

普通高等教育"十二五"规划教材

Fundamentals of Digital Electronics

潘 松 陈 龙 黄继业 编著

数字电子技术基础

(第二版)



- 讲技术，授技能，求职就业的帮手
- 布情境，述过程，教学改革的能手
- 举示例，重实践，能力培养的强手

科学出版社

(TP-6755.0101)

数字电子技术基础（第二版）

本书特点

- 将传统数字技术与现代数字技术有机融合
- 优化教材结构，使课程安排大幅提前
- 信息透明，保持知识结构的合理性和新颖性
- 有利于与其他重要后续课程构成创新能力教学课程体系
- 丰富的习题、实验与设计项目，注重自主创新能力培养
- 以表格形式应用HDL，优化学时安排

本书配套课件、实验示例源程序资料、相关设计项目的参考资料等教学资源可到网站www.kx-soc.com下载

技术分社：http://www.abook.cn
建议上架类别：电子通信　计算机技术


www.sciencep.com
ISBN 978-7-03-041639-1
定 价：42.00元

普通高等教育"十二五"规划教材

数字电子技术基础

（第二版）

潘 松 陈 龙 黄继业 编著

科 学 出 版 社

北 京

内 容 简 介

作为本科数字电子技术的教科书，本教材借鉴了目前国外知名高校同类教材的选材和教学理念，将传统的手工数字技术作为通向现代数字技术的桥梁，在总体上不减少传统和现代数字技术基本内容，且保证教学成效的前提下，最大限度地降低对前期基础知识的依赖，循序渐进地推出该课程所有必须讲授的内容，从而打破教学模式的局限。将目标定位于使学生在数字电子技术的基础理论、实践能力和创新精神三方面有明显的进步。教材将引导学习者基于现代数字技术理论，在全新的软硬件平台上实践已学到的数字技术基础知识，有效提高面向现代数字技术的工程能力，以高起点适应相关后续课程的要求。教材还给出了大量自主设计型实验项目。

相比于同类型的传统教材，本教材的特色突出表现在以下三方面：将传统和现代的教学内容和教学方法有机融为一体；能毫无障碍地在低年级进行教学（如在本科第一学期）；强调并着力培养学生的自主创新能力。

为了方便授课和实践指导，将同时推出与本书各章节内容完全对应的CAI教学课件。

本书可作为本科或高职院校电子工程、通信、工业自动化、计算机应用技术、仪器仪表等专业的专业基础教材，或作为相关专业技术人员的自学参考书。

图书在版编目（CIP）数据

数字电子技术基础/潘松，陈龙，黄继业编著. —2版. —北京：科学出版社，2014
普通高等教育“十二五”规划教材
ISBN 978-7-03-041639-1

Ⅰ.①数… Ⅱ.①潘…②陈…③黄… Ⅲ.①数字电路-电子技术-高等学校-教材 Ⅳ.①TN79

中国版本图书馆CIP数据核字（2014）第187067号

责任编辑：赵卫江 / 责任校对：王万红
责任印制：吕春珉 / 封面设计：曹 来

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
双青印刷厂印刷
科学出版社发行 各地新华书店经销

2008年10月第 一 版 开本：787×1092 1/16
2014年 8月第 二 版 印张：20 3/4
2014年 8月第一次印刷 字数：471 000

定价：42.00元

（如有印装质量问题，我社负责调换〈双青〉）
销售部电话 010-62134988 编辑部电话 010-62138017（HI01）

前　言

近半个世纪以来，电子技术的发展速度是遵循着指数规律的，这对摩尔定律做了最好的诠释。这里的电子技术应该是指半导体集成电路技术，而最能印证摩尔定律的则应属数字电子技术。《实用数字电子技术》的作者 N. P. Cook 认为，第二次世界大战以来，电子学对现代世界的发展所做的贡献超过了所有其他学科。电子工业已超过了汽车和石油工业成为世界上最大的工业，而且此巨型工业的一个重要趋势是从模拟技术向数字技术的转化。数字技术将曾经毫不相干的领域融为一体，导致 90%以上电子产品采用了数字技术，数字电子技术还将继续整合整个工业体系，促进人类在各个不同领域的进步。

显然数字电子技术的发展最能体现现代科技的进步，展示人类不可限量的创造力和永无穷尽的创新追求！

随着数字电子技术的高速发展，40 年前未曾有过的大量学科纷纷出现在高等学校的课堂中，如 DSP 技术、SOC 设计、EDA 技术、数字通信、嵌入式系统、硬件描述语言、面向用户的微电子技术、软件无线电等。这其中许多被列为当今的核心科技学科，自主型科学技术的重心，引领着未来电子技术的发展方向，也预示着前景良好的就业取向。在这些领域的发展中，曾经的 TTL 器件的 5V 工作电压迈向了 0.6V 的芯核电压；曾经风靡全球的仅几兆赫（MHz）主频的 8088 CPU 已变身为跨入 10GHz 门槛的各类高速处理器；过去仅包含数个逻辑门的器件变成了现在特征尺寸达 45nm 的 SOC 系统；一度靠手工设计技术将 74 系列器件组合成“板上系统”的时代也早已脱胎为基于 EDA 技术实现的“片上系统”时代。所有这一切，把作为这一领域的专业基础课——“数字电子技术”的地位和重要性推到了前所未有的高度，同时也对这一课程的教学内容提出了极大的挑战！

然而，难以乐观的是目前国内多数高校该课程所对应的教材的基本内容和实验模式几乎仍停留在 40 年前，并仍以那个年代的手工数字设计技术为核心教学内容与考核内容。其不变的理由正如一位作者所说：虽然新器件、新方法不断涌现，但许多基本理论、基本器件和典型应用是永恒的；另一位作者甚至认为，只要没有走出硅片的范畴，电路的复杂程度和集成规模变化再大，其课程所讲的基本知识、基本理论和基本方法都不会变。

显然，这种认识是严重违背马克思自然辩证法思想基本原则的。

与经典物理学或数学根本不同，现代电子学教学内容的发展变化既是该学科的特点，也是适应技术进步和市场走向的自然选择，特别是数字电子技术更具强烈的时间性。谁又能说远古的砖木取火技术在当时相当长的一段时间内不是最“基本的”？美国 Stanford 大学的 J. F. Wakerly 在其《数字设计原理与实践》一书中说：“如何帮助学生去适应不可避免地要面临的变化，这才是最困难的。因为在这个领域的一般教科书都因摩尔定律而缩短了它的适用期。”

其实，在当今电子技术飞速发展的时代，即使经典原理也未必能幸免被抛弃的命运。美国 Star Bridge Systems 公司曾经采用 FPGA 及 Viva 语言开发出所谓 Hypercomputer 超

级电脑后，对此测试的美国国家航空航天局（NASA）的专家表示："其运行速度无与伦比，这一产品的性能令人过目难忘。"美国 Xilinx 公司的 CEO Willem Roelandts 即认为："由冯·诺依曼提出的电脑架构已经走到了尽头。"我们知道，当今计算机仍在沿用的经典构架之一是 50 多年前由匈牙利数学家冯·诺依曼（John von Neumann）提出的。

再看一个相隔 10 年（远非 40 年）的变化实例：1995 年第二届全国大学生电子设计竞赛和 2005 年的同一赛事中，出现了几乎相同的偏数字技术的赛题，即设计一个输出信号频率能等步长数控的正弦信号发生器。主要不同点为，前者指标是，输出信号上限频率是 20kHz；后者是 10MHz！显然，面对 2005 年的赛题，1995 年曾经成功的设计工具、设计技术、设计方案、系统结构、硬件实现、乃至部分设计理论都用不上了。这意味着 10 年前曾经是优秀的电子设计工程师，如果不随时代更新其知识，10 年后只能面临被淘汰的命运！

显然，脱离了科技进步和时代背景的"数字电子技术"是没有任何实际意义的！

然而遗憾的是，目前国内高校的数字电子技术基础课的绝大部分教材内容和教学模式仍然停留在 20 世纪 70 年代末。这一现状严重影响了应有的教学质量和就业竞争力。

一、 目前教学内容、 教学模式和教学目标存在的基本问题

1. 过于偏重理论教学而忽视实践能力的培养

暂且不提其"理论基础"是否存在许多值得商榷的内容，仅其学习模式和学时安排就有悖于此课程的基本性质——"技术基础"。调查表明大多数学校对此课程的学时安排是 64 学时授课、16 学时实验，而且以验证性实验为主。显然，实践要求被严重弱化，而且现有趋势表明，此课程已基本退化为依赖题海战术的普通基础课，诸如搞试题库、习题集、标准答案、联校统考等。殊不知这一切恰恰埋没了这门作为当今电子科技领域最富变化、最具活力、最贴近实践、最需创新能力的学科的鲜明特质，从而偏离了此课程真正的教学目标和教学要求。其结果是造就了大批只擅长纸面答题而畏惧动手实践的学习者，严重影响了后续课程的学习和求实创新精神的培养。事实上，作为一门面向技术的主干必修课，它既是许多重要后续课程的基础课，同时又理应成为培养学生尊重实践、勇于探索、积极创新等优良素质的启蒙学科。现代电子与计算机领域中拥有重大经济价值的自主创新项目多产生于数字电子技术领域。显然，数字电子技术的教学应该是呵护和激发创新精神的源头，这个领域不需要什么习题集，也永远没有标准答案，它提倡个性、鼓励想象、适应变革、崇尚实践！

2. 陈旧的教学内容

那些无法与现代数字技术接轨的陈旧教学内容，主要包括对传统组合电路和时序电路的分析与设计的纯手工技术，及数十年不变地围绕这一课程的核心教学内容，就如同一包

内容不变而生产日期不断被更新的过时食品。这种“食品”对应的现象是，所有这些似乎与教材的不断再版和更新无关，教材从不直白其内容实为仅适合于低速小规模数字电路设计的“手工数字技术”，也没有任何文字坦承将要传授给学生的不过是诞生于 20 世纪 60 年代、成熟于 70 年代、完全淘汰于 80 年代初的陈旧的内容。

这无疑会在读者中产生许多误导，例如会误将过时的手工技术、分析方法和设计流程当成现代数字技术基础知识去应用；如将逻辑化简误当作系统优化的目标去追求；或将逻辑功能手工分析方法误认为这就是现代数字系统的时序分析；甚至会将这些教材中仅适合于低速小规模条件下的数字知识当成一般数字技术误用到高速大规模逻辑设计中。

至于将 IEEE/ANSI 曾于 1984 年制定的，并很快在国际上不再使用的逻辑图形符号标准一直在我国教科书中作为“国家标准”推广至今，则是另一明证。其实这种早已过时的所谓国标符号，于 20 世纪 80 年代后期就被数字技术业界升级至 ANSI/IEEE 1991 标准(Distinctive Shape Symbols)。其结果是，迫使已完成现在的数字电路课程学习的学生，不得不重新学习和熟悉 ANSI/IEEE 1991 标准的逻辑符号，才能面对实用的电子工程技术。

3. 脱离工程实际

目前多数数字电子技术教学脱离工程实际，甚至充斥着偏离工程实际的伪命题和伪技术。例如，当今的 TTL 74LS 系列器件原本可毫无问题地与 CMOS 器件相互接口，甚至 3.3V I/O 电平的器件也能可靠地接口 TTL 和 CMOS 器件，然而现有教材中无不推荐使用诸如要加上拉电阻、三极管电路或是插入专用驱动器件（如 40109）等多种所谓“接口技术”，来解决这些无中生有脱离实际的“电平不匹配”命题；或者介绍要用三个 CMOS 驱动一个 TTL 器件之类的夸张“技术”；再如对于毛刺脉冲的解决方案，工程上原本应该选择时序电路，然而有的教材却罗列了许多脱离实际甚至错误的方法，如改变逻辑结构的“冗余技术”、端口接电容的滤波技术等；更有甚者，有的教材单纯地介绍只要获得对应的反馈逻辑函数，就能通过控制通用逻辑器件的清零或置数端，构建成指定进制的计数器，却从不提及或讨论，以此类方法设计的逻辑系统在外部温度或电磁环境等因素发生变化时，控制逻辑信号是否会出现毛刺脉冲，从而提前启动清零或置数，例如，74LS161 理论上的十二进制计数设计很可能由于外部因素而变成八进制计数器！

4. 知识体系的结构性缺陷

传统教材知识体系的结构性缺陷主要表现在以下三个方面：

（1）其知识体系不具备以此及彼的可推广性和一般性。例如无法将逻辑电路的分析方法应用到实际工程上；又如设计时序电路的经典五步骤其实只适用于简单电路（如模 7 计数器等）的设计，根本无法推广到更实用的诸如模 70、模 700 乃至模 7000 等多变量的同类计数器的设计，从而约束了学生的创造力。

（2）仅罗列枝节性知识，没有将这些知识上升为一般性理论和实践依据。例如，罗列

了各种时序电路应用模块（如各类计数器），给出了诸如状态变量、状态图、状态表等重要概念，却始终没有迈上“一切同步时序电路都是状态机的特殊形式”的关键台阶。所导致的后果是，多数学习者除了仅能完成基于笔头的“书面实践”外，无法再向前多跨一步。

其实这种教学安排本身就违背了人类的基本认识规律。人类对自然的认识规律是从特殊到一般，再从一般回到特殊的过程；而人对自然的能动作用则体现在后一阶段。如果教学中只罗列各类计数器的类型和设计，却没有将其升华为更具一般意义的理论层次，即状态机及其应用方法（国外许多同类教材都介绍状态机的设计），无疑是舍本求末之举。

(3) 缺乏前后呼应的一致性。尽管多数教材都安排了存储器、模/数、数/模等实用器件的介绍，但都仅仅限于对它们的结构和原理的介绍，却不介绍它们的实用方法，更没有相关的实验安排，导致与主干内容亦呈游离状态，使得本来以推介“技术”为目标的教科书沦为器件说明书。其原因只有一个，即疏漏了对状态机设计和应用等核心知识的介绍！

5. 技术理念落后

现有教材的核心内容始终是围绕着低速小规模电路的技术理念展开的。例如，甚至在最新版的权威教材中仍有这样的文字：“在数字系统中，常常需要用到一些脉冲信号的产生和变换电路”，“单稳态电路也是数字系统中常用的一种脉冲整形电路”。事实上，即使现代最小规模的数字系统设计中也不会包含任何脉冲信号变换或诸如单稳态类涉及阻容元件的电路，这是因为现代数字系统的普通工作频率都高达数十至数百兆赫，而涉及阻容元件模拟电路的脉冲变换或单稳态电路的频率通常无法超过 10^2 kHz 量级。一个如此低频率的电路怎么可能出现在现代数字系统中呢？殊不知现代数字系统与模拟电路已是两个分离的世界，它们被独立研究，单独设计，它们之间的联系只能是 A/D 和 D/A。

6. 教学目标定位过低

在长期延续下来的低速小规模理念的引导下，数字电路教学过低的教学目标定位是必然的。大多数教学情况除了满足于纯笔头的书面考核外，其动手能力的训练基本仍停留在几十年一贯制的基于 74 系列器件的译码器、计数器、抢答器或诸如交通灯控制、电子钟这类低层次简单设计上。这势必使学生以极低的起点和过窄的视野去面对大量与之联系紧密的重要的后续课程的学习。

相比于此领域的教学改革先行者的成就，其间的差距是巨大的。例如国内已有不少院校（包括我校多数院系）将数字电路课程放在本科第一或第二学期，实践训练的内容包括超过数万至数十万逻辑门规模的数字系统自主设计训练，不少受益的学生在各类电子设计竞赛中获得了好成绩；又如美国 Stanford 大学将数字电路课 Digital System (1) 也放在本科第二学期；清华大学电子系本科生从一入学就人手获得一块 Altera FPGA 实验开发板，该校计算机专业本科二年级学生就能自主设计出各种极具创新特色的数字系统；美国 Michigan 大学本科一年级学生就能设计数字电子琴，其中 FPGA 控制 VGA 显示五线谱，

PS/2键盘作为琴键；东南大学在一次省级数字电路课程电子设计竞赛中，有一组同学完成了指纹识别数字锁的设计而获一等奖。

7. 课程安排的时间太晚

目前数字电路课程绝大多数情况下仍被放在本科的第四甚至第五学期。这在40年前没有什么不妥，因为那时数字技术起步不久，还没有什么上规模的数字产品，与数字技术相关的课程也非常少，因此那时的数字电路课对后续课程或就业的影响甚微。此外，那时的数字技术尚处于低速小规模手工技术阶段，这使得数字电路与模拟电路常同时出现于同一电路结构中，从而要求学习者较多地关注脉冲及数字信号的产生与处理上（而非现代数字技术所追求的功能实现和系统优化上），这就难免要求学习者具备更多的数学知识、半导体器件知识和模拟电子线路的基础知识，于是使得此课程根本无法提前安排。

但在今天，随着基于数字技术基础的大量专业技术类课程的爆炸式涌现，该课程已成为许多重要后续课程必不可少的基础课。然而第四、五学期过晚的课程安排导致余下过短的时间（通常不到一年）使学生难以应付大量的后续课程的加入，更不可能有足够的时间通过有效的实践来消化它们，学习效果自然大打折扣。而这些课程又多数与未来的就业或可能的深造关系密切。实际上，改变这种状态的关键是应认识到，当今是数字技术和系统集成的时代，对于数字技术的学习，完全可以暂时把模拟电子技术及其理论暂时先放在一边，从而有条件将此课程的教学提前几个学期进行。

二、本教材的特点

鉴于以上讨论，并针对数字电路传统教材和教学中存在的问题，本书在内容编排与选材、实验内容与模式、课程设置和学习目标定位上，突出了以下特点。

1. 将传统数字技术与现代数字技术有机融合

事实上，为了改善教材内容过于陈旧的现状，不少新版教材也增加了诸如硬件描述语言、FPGA应用、EDA软件应用等内容。但问题在于，除了生硬地增加几个章节外，并没能将传统与现代的内容之间进行“无缝接轨”，即在传统教学内容中看不到迈向现代技术的任何脉络痕迹和铺垫，而在加入的现代数字技术章节中也体现不出对前期教学内容的传承性、延续性和奠基性，其结果除了增加更多的学时外恐怕就是学习者的迷茫了。

有鉴于此，本教材的做法是，在介绍组合电路（第4章）的大部分内容前仍旧按照传统的方法，介绍了组合电路的性质、分析方法和设计技术等，再给出一系列逻辑器件的功能用法的说明。但在最后，将前文所列的诸如编译器、编码器、加法器等器件做一归纳和升化，提出任何组合电路的本质是相同的，只要用一张能表达它们输入输出关系的真值表即可描述，于是便引入了广义译码器概念，使读者对组合电路的认识和处理方法上升到一个一般化的层面，从而能在第6章的Quartus II软件应用介绍中，顺理成章地将之前所学

的知识融入到现代数字技术的组合电路设计与分析中。同理，在第 7 章的时序电路基本内容部分，特别是在对各类计数器及相关集成器件功能的分析设计的介绍后，将它们抽象为传统教材中不曾有的万变不离其中的同步时序逻辑电路的一般模型，其实它是由一个广义译码器和一个简单的寄存器构成的，从而将基于传统方式繁复而不具一般意义的时序电路设计回归为一个普通译码器的设计。

再前进一步，第 8 章展示了对传统数字技术和理念的升华，即最后证明了一切同步时序电路，包括计数器都不过是状态机的特殊形式；进而证明，构成状态机中的广义译码器本质上就是一个状态译码器，而这部分电路模块是现代自动设计技术中最容易解决的部分！由此便水到渠成地引导读者学会了利用传统知识来设计各类有限状态机，乃至任何形式的组合与时序电路功能模块。例如，学会利用状态机控制各类熟悉的对象，如步进电机、电饭煲或 A/D 采样等，这些内容成为传统教材中不可能出现的实验项目，达到了将传统手工数字技术与现代自动化数字技术的基础理论和工程实践有机融合的目的，实现了从传统数字技术向现代数字技术的平滑过渡！

为了能在第 8 章中，根据前文的铺垫给出基于现代数字电路自动设计技术的软硬件设计与分析方法，及时推出相关的硬件实验项目，同时节省了学时数，在第 4 章和第 6 章（此章也介绍了触发器）中根据前文的知识铺垫及时穿插着给出了 PLD 结构的内容。

2. 优化教材结构，使课程能大幅提前

根据以上的讨论，本科数字电子技术课程的大幅提前具有极端重要性。从我校已经历的至少五个成功的该课程的教学试点周期（提前至本科第二甚至第一学期）以及参与该教学试点的学生后期学业与能力培养来看，也具有良好的可行性。据此，在总体上不减少传统和现代数字技术必备的基本内容，且保证教学成效的前提下，本教材在前期知识中大幅减少和推迟逻辑门底层电路结构和脉冲电路（放在第 11 章）的介绍，最大限度地降低对前期基础知识的依赖（如高等数学、模拟电子线路等）。在学习和实践过程中允许保留无关大局的疑问和不解，因为这可留待在后续相关课程中带着问题来学习和解惑，同时也鼓励并安排学生从课本以外的资料中寻求答案，解决问题，开拓视野。事实上，现在多数国外同类教材都不对门电路的结构做过多的深入说明（至多介绍器件的电气特性）。因为在现代电子系统设计中，对于深入底层结构的理解与否并不会影响系统集成的设计水平，更何况后续的电路分析和模拟电子线路课程能做很好的补充。这一安排的另一好处是，大幅减少了教学课时数，有利于将更多的课时数让位于实验与工程设计训练。

3. 信息透明，保持知识结构的合理性和新颖性

鉴于电子技术类知识的强烈时效性，凡有必要处，都将毫不掩饰地指明教材中相关内容的历史背景和适用范围，随时说明这些内容属于阶段过渡性质的还是既定的学习目标，哪些理论和技术适应于低速小规模条件，哪些方法或器件仅在历史上是常用的；清晰说明手工技术与现代自动技术间、数字系统的传统分析方法与现代分析方法间的区别等，使读

者能及时调整自己的关注重心。尽管本教材中也大量介绍了传统（手工）数字技术的内容，但这只是为最终目标服务的，即引导读者基于新的数字技术理念，在全新的软硬件平台上强化数字技术基础知识的学习效果。此外在存储器（第 9 章）和 A/D、D/A（第 10 章）章节中安排了许多针对性强的应用实例和相关的自主创新型综合实践项目，这种安排在传统教材中是不存在的，但却是必不可少的。

加拿大多伦多大学数字电路教材《数字逻辑基础》的作者 Stephen Brown 说："虽然现代设计人员已经不再使用手工设计技术了，但手工设计可以让学生直观地了解数字电路是如何工作的。可以让学生深入了解自动化设计技术的优点。"但他同时指出其教材的重点是介绍自动设计技术，这与本教材的结构安排是吻合的。

4. 有利于与其他重要后续课程构成创新能力教学课程体系

基础理论和技能的教学固然重要，但培养学生的自主学习能力，激发学生的创新思维，鼓励学生的创造性实践则更重要，这是授人以渔的久长之计！因此，作者并没有将本教材作为一本孤立的数字电子技术基础课程的课本来构建，而是兼顾其后续课程的衔接，包括基础知识的衔接、实验内容的过渡、设计项目的可延伸性，以及对创新能力培养的铺垫等，尽可能为后续课程营建良好的接口。由此便可将数字电路、单片机技术、EDA 技术、SOC、嵌入式系统等具有较大相关性的课程构建为一个课程系列有机体。这个体系最早可以从本科第一学期开始，在完成数字电路课程后，相关课程紧随其后。无疑，这样的课程设置将比传统情况整整提前约三个学期，其意义显然是重大的。这可以优化相关专业的课程设置，让学生提前进入理论与工程实际相结合的高效学习和训练阶段，提前激发创造欲望，提前具备进入自主设计性空间的能力，提前为未来的学习和实践打开充裕的时间空间、自主学习空间和就业准备空间。我校在这方面的教学试点和现在的大规模推广已经给出了有说服力的证明，例如相比于其他同类情况，那些曾经参加这一教学体系的学生在大学生电子设计竞赛、飞思卡尔车模大赛，以及一些企业赞助的自主设计赛事中都获得了更多的奖项和更好的成绩。

5. 注重自主创新能力培养

本教材的目标定位有两个：①现代数字电子技术基础知识的传授和对应的实践能力的培养；②通过教学的启迪和教材中大量的有创意的实验项目训练，能动地激发创新意识，培养自主创新能力。从而使学生在数字电子技术的基础理论、实践能力和创新精神三方面能得到同步收获，有能力提早进入大学生课外科技活动。

本书以数字电路手工设计技术的介绍作为跨上进一步学习的台阶，以自动设计技术的学习为能力培养的手段，注重现代数字技术基本知识、理论和方法的介绍，注重工程能力、分析能力和实践能力的培养，全书构建了一个从介绍基础知识向创新能力培养逐级递进的学习和实践的阶梯；特别是介绍了状态机构建和数字系统自动设计软硬件后，为读者在这一领域自主设计能力的发挥和创新思路的验证提供了武器和实现平台。

6. 以表格形式应用HDL，优化学时安排

本教材在推出广义译码器和计数器一般模型的基础上证明，一般数字电路的最核心的功能模块是广义译码器，其实就是组合电路模块；而组合电路模块的表述和构建的关键就是一张真值表，即只要获得了表述组合电路功能的真值表，就能通过数字系统自动设计软件完成对应的组合电路的设计与测试。由此不难发现仅使用 HDL 的 case 语句来表述这个真值表，就能顺利解决 HDL 教学费时的问题。于是在教材中，对于 HDL 仅以“表格”方式利用极少数语句实现关键部件的设计，而完全不必费时费力地去专门学习 HDL 知识，从而聚焦重点知识，注重系统设计，着眼能力培养，极大缩短了授课的学时数。

本教材推荐的授课课时数仅约 40 学时，实验课时数则不少于 32 学时，其中验证性实验约 15%，其余为自主设计与创新型实验，且建议另外再增加配套的实践环节和实验课时数。读者在完成本书的学习及推荐的实验，特别是完成了最后一章，即第 12 章的数字系统综合设计项目，以及相关联的课程群体系（除了本教程推荐的内容外，紧接此后的这个课程群体系的其他内容大致包括单片机技术、SOPC 技术、DSP 技术、EDA 技术、计算机组成与设计、计算机接口技术、嵌入式系统等）后，有理由相信已具备了参加全国大学生电子设计竞赛等课外科技活动的知识和实践能力，而这时最多刚完成二年级的学习！事实证明，参与了这一课程体系教学试点的多数学生，本科四年后的就业或升学竞争力一点也不亚于传统情况下的硕士生。

与本书配套的 PPT 教学课件、实验指导课件、实验源程序，以及 Mif Maker 应用软件等资料可以免费索取，可浏览科学出版社网址 www. abook. cn，或康芯电子有限公司网址 www. kx-soc. com，欢迎就数字技术教学改革进行交流探讨，作者 E-mail：chenlong@hdu. edu. cn，hjynet@163. com，pmr123@sina. cn。

作 者

2014 年 6 月

于杭州电子科技大学

目　录

第1章 数制与编码

不同进制的数制及其表达方式、运算方式，以及编码方式是数字技术中最基础和最重要的内容之一。本章首先介绍模拟信号与数字信号的基本概念和数字信号的特点，然后介绍一些常用的数制及其表述方法，以及不同数制间的转换与运算方法，最后介绍编码的概念与不同的编码格式。

1.1 模拟信号与数字信号

在自然界中存在着各种各样的物理量，但就其特点和变化规律而言，主要可分为两大类。一类物理量在变化时间上和数量上都是离散的，这类物理量称为数字量。而另一类物理量在变化时间上和数量上都是连续的，这类物理量称为模拟量。在电子电路中，按照所处理的信号形式，可以将电路分为模拟电路和数字电路。模拟电路处理模拟信号，数字电路处理数字信号。

1.1.1 模拟信号与数字信号的概念

模拟（Analog）信号是指该信号的幅度量值随着时间的延续（变化）而发生连续变化的信号。例如随着时间的延续，温度可变高或变低，速度可变快或变慢，声音可变强或变弱，等等。其特点是，它们的变化总是连续的，平滑的，且在任何一个极短的时间段内都不可能发生突变。如果利用传感器把它们转变为电信号，则这些电信号的幅度（电压或电流）也是随时间连续变化的。

用以传递、加工和处理模拟信号的电子电路被称为模拟电路。模拟电路的具体功能是对模拟信号进行放大、滤波、运算与处理等。

数字（Digital）信号是指该信号的幅度量值随着时间的延续（变化）而发生不连续的、具有离散特性变化的信号。数字信号通常只有两种相互对立的状态，如开关的打开与关闭，导线的连接与断开，电压的高和低，信号的有和无，等等。用于处理数字信号的电路，如传送、存储、变换、算术运算和逻辑运算等的电路称为数字电路。

由于拥有许多优秀的特点，数字电路被广泛应用于计算机、通信、工程控制、测量仪器、信号处理和图像显示等方面。

1.1.2 数字电路与模拟电路的区别

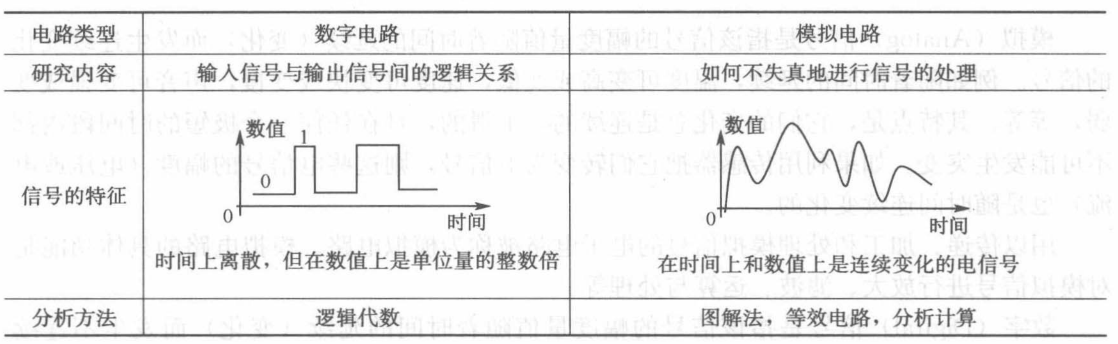
与模拟信号不同，数字信号的确立具有一定的人为性。模拟信号比较直接地反映了自然界中真实的物理量的变化，而数字信号量则是通过人为的选择，比较间接地反映实际的物理量的变化。例如在任一小段的时间段内，连续变化的电压值的取值具有无限多种，如果用计算机来直接处理这个电压模拟量显然是不可能的。因为再大的计算机内存也无法放下在此时间段内所发生的，而且是实际存在的无限多的电压值。因此必须将随这段时间变化的电压连续值进行抽样离散化，变成有限数目的离散化了的电压值，即能转换成可用计算机直接处理的数制数据，如二进制数。

数字电路也称逻辑电路，尽管它处理的数值通常用数码 1 和 0 来表示，但其含义根据不同的情况可以有不同的解释。例如，可以将 0 和 1 看成实际的数据，对由此构成的二进制数进行加减乘除的算术运算；但也可以用 1 和 0 来表示逻辑的“真”和“伪”、“是”和“否”的判断，因此可以通过对数据 1 和 0 的处理来实现对逻辑的电路运算，或称逻辑操作；甚至还可以用 1 和 0 来表达数据的正负。显然所有这一切，普通的模拟电路及相关数据是难以实现的。

在数字电路中通常规定高电平为逻辑 1，或数值 1，例如可以规定大于等于 2.7V 的电压值为高电平；规定低电平为逻辑 0，或数值 0，例如规定小于等于 0.5V 的电压值为低电平。这种表示法称为正逻辑。反之，若规定低电平为逻辑 1，高电平为逻辑 0 的表示法称为负逻辑。上述的 2.7V 和 0.5V 通常称为判别高低电平（逻辑 1 或 0）的阈值。

表 1-1 列出了数字电路与模拟电路的主要区别。

表 1-1 数字电路与模拟电路的主要区别



电路类型	数字电路	模拟电路
研究内容	输入信号与输出信号间的逻辑关系	如何不失真地进行信号的处理
信号的特征	数值 1 0 0 时间 时间上离散，但在数值上是单位量的整数倍	数值 0 时间 在时间上和数值上是连续变化的电信号
分析方法	逻辑代数	图解法，等效电路，分析计算

1.1.3 数字电路的特点

数字电路主要采用二值逻辑（或二值数据）。数字电路的理论基础和数学工具是逻辑代数。在数字电路中的电子器件，如二极管、三极管和 MOS 管等主要工作在开关状态，即导通或截止两种状态之一。对电路中电压和电流的精确值要求并不高，只要电路能可靠地区分高、低电平即可。例如，某信号端口的输出电压在外部环境的影响下，在 3.3V 与

4.7V之间变化，从数字电路角度看，其输出的数据是稳定的精确的，因为始终是高电平1。这也表明，数字电路的抗干扰能力远好于模拟电路。

数字单元电路（门电路、触发器等）结构简单，便于集成化。数字电路功能的表示方法主要有真值表、卡诺图、逻辑表达式和波形图等。

在电路结构上，数字电路和模拟电路都是由晶体管、集成电路等电子组件所组成。但与模拟电路相比，数字电路具有以下优点：

(1) **稳定性好，抗干扰能力强**。在数字系统中，数字电路只需判别输入、输出信号是逻辑1还是逻辑0，而无需知道它们所表示的电压或电流的精确值。只要噪声信号不超过高低电平的阈值，就不会影响逻辑状态，因而具有较好的稳定性和抗干扰能力。

(2) **容易设计，便于构成大规模集成电路**。数字电路中的晶体管均工作在开关状态，只用“通”和“断”两种不同的状态来代表二值信息1和0。大多数数字电路都可以采用集成电路来系列化生产，成本低廉，使用方便，从而进一步促进了集成电路在数字电路中的广泛应用。

(3) **信息的处理能力强**。数字系统能方便地与电子计算机连接，利用计算机的强大功能进行数据处理，对输入的数字信号进行算术运算、逻辑运算，还可具有逻辑推理和判断的能力，并实现实时控制。

(4) **精度高**。可以很容易地通过增加二进制位数，使数字电路处理数字的结果达到人们所希望的精度。因此，数字电路组成的数字系统工作准确，精度高。

(5) **保真度好**。这也是稳定性好的一个方面。信号一旦数字化以后，在传输和处理过程中信息的精度不会降低，即结果再现性好。例如同类型的数字电路总是能精确地产生相同的结果，而模拟电路输出的电压和电流信号则会由于受组件参数的变化和周围环境温度、湿度的变化，偏离原有的量值，从而产生不同的结果，这就是信号的失真。

(6) **便于存储**。利用数字存储器可方便地对数字信号进行保存、传输和精确再现。

(7) **便于利用自动化设计技术**。高度发展的计算机技术使得数字系统设计完全可以利用自动化设计技术来实现。即直接利用计算机来完成数字电路系统的硬件设计与测试，从而极大提高了数字电路的设计效率，这是模拟电路设计难以企及的。

(8) **功耗小**。由于数字电路中的组件均处于开关状态，极大地降低了静态功耗。

数字系统，或数字电子系统是处理数字信号的电路系统，它通常是由不同功能的数字电路模块构成的（计算机也可以认为是一个数字系统），它们具有模拟电路无法比拟的优势，其应用日益广泛，在整个电路系统中所占的比重也越来越大。随着集成电路制造技术的不断进步和发展，数字系统正在向低功耗、低电压、高速度、高集成度方向迅猛发展。集成电路的成本不断降低，产品类型层出不穷，大量使用大规模数字逻辑功能模块来高效设计数字产品已成为现实。因此在电子信息、通信与计算机工程领域，数字电子技术是一门发展最快、跨学科门类最多、应用最广的学科和实用技术。

1.2 数字系统中的数制

人们在长期的生产实践中发明和积累了多种不同的计数方法，既可以用不同的数码表

示不同数量的大小，又可以用不同的数码表示不同的事物或同一事物的不同状态。

在表示数量大小时，仅用一位数往往会不够用，这就需要用多位数来表示。在多位数码中，每一位的构成及从低位向高位进位的规则称为进位计数制，简称数制。在数字系统中常用的进位计数制有十进制、二进制和十六进制，有时也会用到八进制。

1.2.1　十进制数表述方法

十进制数是日常生活中最常用的计数体制，它采用10个数码0、1、2、3、4、5、6、7、8、9，按一定规律排列起来表示数值的大小。在多位数中，低位和相邻的高位之间的进位关系是"逢十进一"，这就是十进制的含义。

十进制的表示常用下标10、D或缺省不作任何标记。如十进制数的78可以表示为：78_{10}，78_D，78D或78。多位数中不同位置上的1代表的数量大小称为这一位的"位权"，简称为权。整数部分从低位到高位的权依次为，10^0，10^1，10^2，…。小数部分从高到低位的权依次为10^{-1}，10^{-2}，10^{-3}，…。因此，一个多位数可以表示为每一位乘以它的权值，然后相加。例如，十进制数358.67可以写成

$$358.67=3\times10^2+5\times10^1+8\times10^0+6\times10^{-1}+7\times10^{-2}$$

从以上表示方法可以看出十进制数具有以下特点：

(1) 在每个位置只能出现（十进制数的）10个数码中的一个。通常把数码的个数称为基数。十进制数的基数为10。

(2) 低位到相邻高位的进位规则是"逢十进一"，故称为十进制。

(3) 同一数码在不同的位置（数位）表示的数值是不同的。一般把不同位置上数码的单位数值称为位权或权，而把每个数位对应的数码称为系数，则某数的数值（位值）等于该位系数与权的乘积。这样，任何一个十进制数N都可按权展开为

$$\begin{aligned}(N)_{10}&=a_{n-1}10^{n-1}+\cdots+a_1 10^1+a_0 10^0+a_{-1}10^{-1}+\cdots a_{-m}10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times10^i\end{aligned}\qquad(1\text{-}1)$$

式中$(N)_{10}$为十进制数，10表示基数，a_i称为第i位的系数，是10个数0～9中的一个。n是整数部分的位数，m是小数部分的位数。

小数点用来区分一个数的整数和小数部分。更准确地讲，相对于小数点不同位置所含权的大小可用10的幂表示。也就是说，十进制数各位的权值为10^i，i是各数位的序号。例如，十进制数358.67，处于不同位置的数字符号代表着不同的意义，即有不同的权值。最左边的位权最大，称之为最高有效位（MSB），最右边的位权最小，称之为最低有效位（LSB）。

为区别不同数制表示的数，通常用括号外的下标字母表示括号内的数制，以作强调说明。如十进制数用D或10表示；二进制数用B或2表示；十六进制数用H表示等。

如$(N)_{10}$，$(418)_{10}$，$(11011.101)_2$，$(89A.4)_H$，$(1101)_B$，6BH等。

一般情况中，由于习惯和默认，或在简化表述下，省略了数制下标。如默认1101.11、782.45分别表示二进制和十进制数。具体情况可联系上下文来判别其数制。

1.2.2 二进制数表述方法

在数字电子系统中，或计算机运算中，十进制数不便于直接表述与计算。二进制则成为数字电路中最常用的计数体制。在二进制中仅采用 0 和 1 两个数码来表述，因此其基数是 2，低位到相邻高位的进位规则是“逢二进一”。

类似于十进制数，对于任一个二进制数 N，其按权展开式可表示为

$$(N)_2 = a_{n-1}2^{n-1} + \cdots + a_1 2^1 + a_0 2^0 + a_{-1}2^{-1} + \cdots a_{-m}2^{-m}$$
$$= \sum_{i=-m}^{n-1} a_i \times 2^i \tag{1-2}$$

式中 $(N)_2$ 为二进制数，n 表示整数部分的位数，m 表示小数部分的位数，2 表示基数，2^i 为第 i 位的权，a_i 为第 i 位的系数，$a_i \times 2^i$ 称为位值。

如将 $(11010.101)_2$ 写成权展开式为

$$(11010.101)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

二进制数只有 0 和 1 两个数字，基数是 2，最大的数字是 1，逢二进位，各位的权为以 2 为底的幂。例如，以上的（11010.101）各位的权自左至右依次为 2^4、2^3、2^2、2^1、2^0、2^{-1}、2^{-2}、2^{-3}。

二进制数的算术四则运算规则（表 1-2）与十进制数类似。在计算机中，引入补码表示后，加上一些控制逻辑，利用加法就可以实现二进制的减法、乘法和除法运算。

表 1-2 二进制算术运算规则

加法规则	减法规则	乘法规则
0+0=0	0−0=0	0×0=0
1+0=1	0−1=1（有借位）	1×0=0
0+1=1	1−0=1	0×1=0
1+1=10	1−1=0	1×1=1

以上的二进制算术规则与十进制的算术规则是类似的，只是在进位时按照“逢二进一”，而在借位时则“以一当二”。在进行两个多位的二进制数相乘时，同样可以用类似十进制数运算的左移相加的方式来完成。

同样，二进制的除法规则与十进制的除法规则也类似。如有以下二进制数除法：

$$11110 \div 101 = 110$$

同样可以用以下算式完成：

```
        110
101 ) 11110
      101
      ---
       101
       101
       ---
         0
```

二进制的主要优点可归纳为如下两点：

(1) 二进制只有 0 和 1 两个数字，很容易表示。电压的高和低、晶体管的截止与饱等都可以表示为 0 和 1 两种状态。

（2）二进制数的每一位只有0和1两个状态，只需要设备的两种状态就能表示。所以采用二进制数能节省设备资源。由于状态简单，所以抗干扰力强，可靠性高。

与十进制比，二进制的主要缺点是数位太长，不便于阅读和书写。许多情况下，使用起来也不方便。为此，常用十六进制作为二进制的缩写方式或表述方式。

为了适应人们的习惯，通常在计算机内都采用二进制数，输入和输出采用十进制数，由计算机自动完成二进制与十进制之间的相互转换。

1.2.3 十六进制数表述方法

尽管二进制数在计算机系统中处理起来十分方便，但当直接面对位数很多的二进制数时，计算机程序员就会感到难记难写难认。为了解决这个问题，通常将二进制数用十六进制来表示，因此十六进制成为计算机应用技术中被广泛使用的一种数制。

十六进制数采用16个数码0、1、2、3、4、5、6、7、8、9和A、B、C、D、E、F。其中A～F分别对应于十进制数的10～15，即A表示10、B表示11、C表示12、D表示13、E表示14、F表示15。因此十六进制的基数为16，低位到相邻高位的进位规则是"逢十六进一"。同样，对任一个十六进制数N，其权展开式可表示为

$$
\begin{aligned}
(N)_{16} &= a_{n-1}(16)^{n-1} + \cdots + a_1(16)^1 + a_0(16)^0 + a_{-1}(16)^{-1} + \cdots + a_{-m}(16)^{-m} \\
&= \sum_{i=-m}^{n-1} a_i \times (16)^i \qquad (1\text{-}3)
\end{aligned}
$$

式中$(N)_{16}$为十六进制数，n是整数部分的位数，m是小数部分的位数，16是基数，16^i为第i位的权，a_i为第i位的系数，$a_i \times 16^i$称为位值。如$(7F9)_{16}$的权展开式为

$$(7F9)_{16} = 7 \times 16^2 + F \times 16^1 + 9 \times 16^0$$

1.2.4 八进制数表述方法

八进制与十六进制计数系统一样，在计算机应用中也较常用。在进行特定的计算机编程时，常用到此计数系统，因为它也能够很容易地与二进制系统相互转化。此外，八进制还提供了一种有效的方式来表示较大的二进制数。

八进制数的基数是8，它有0、1、2、3、4、5、6、7共8个有效数码。八进制数用下标8或O表示，如$(617)_8$，$(521)_O$等。对照公式（1-1），八进制的按权展开式为

$$
\begin{aligned}
(N)_8 &= a_{n-1}8^{n-1} + \cdots + a_1 8^1 + a_0 8^0 + a_{-1}8^{-1} + \cdots a_{-m}8^{-m} \\
&= \sum_{i=-m}^{n-1} a_i \times 8^i \qquad (1\text{-}4)
\end{aligned}
$$

式中，n是整数部分的位数，m是小数部分的位数，a_i是数码0～7中的一个。

1.3 不同数制间的转换

本节主要介绍十六进制数与其他数制之间，十进制与二进制数之间的转换方法，以及将一个数从一种数制转换到其他数制的转换方法。

1.3.1 十六进制、二进制数与十进制数间的转换

十六进制数可以用作二进制数的一种缩写方式，因为 4 位二进制数有 16 种组合，可以对应十进制数的 0～15。二进制数转换为十六进制数的方法是：从小数点开始向左按 4 位分节，最高位和低位不足 4 位时，添 0 补足 4 位分节，然后用一个等值的十六进制数代换。

反过来，十六进制转换成二进制的方法是：将每个十六进制数用 4 位二进制来书写，其小数点最左侧（整数部分）或小数点最右侧（小数部分）的 0 可以省去。

相比之下，十进制数转换为二进制数稍嫌麻烦。通常采用基数乘除法。整数部分和小数部分分别转换，最后将两部分合起来，即为所转换的二进制数。

反之，若要将二进制数或十六进制数转换为十进制数，只需将对应的二、十六进制数按各位权展开，并把各位值相加，即得相应的十进制数。

【例 1-1】 将二进制数 $(110101.101)_2$ 转换为十进制数。

解：
$$\begin{aligned}(110101.101)_2 &= 1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}\\&\quad+0\times2^{-2}+1\times2^{-3}\\&=32+16+0+4+0+1+0.5+0+0.125=(53.625)_D\end{aligned}$$

【例 1-2】 将十六进制数 $(4E5.8)_H$ 转换为十进制数。

解：
$$\begin{aligned}(4E5.8)_H &= 4\times(16)^2+E\times(16)^1+5\times(16)^0+8\times(16)^{-1}\\&=4\times256+14\times16+5\times1+8\times(1/16)=(1253.5)_D\end{aligned}$$

1.3.2 十进制数转换为二进制、十六进制数

若将十进制数转换为二进制或十六进制数，可对整数部分和小数部分分别进行转换，最后将结果进行合并。整数部分转换，采用除基数反序取余法。即除以二进制数的基数 2 取余数，再将其商除以 2 取余数。重复这一过程直到商为 0 为止。第一次的余数为二进制数的最低位，依次递增，最后一次余数为二进制数的最高位。

小数部分转换是采用乘基数顺序取整法。即乘以基数 2 取整数，将余下的小数再乘以 2 取整数，直到所需精度为止。小数部分的转换可能出现无限循环或无限不循环情况。第一次的整数为二进制小数的最高位，依次递减，最后的整数为二进制小数的最低位。

【例 1-3】 将 $(59.625)_D$ 转换为二进制数。

解：

整数部分

除法	余数	
2∣59		
2∣29	……… 1	低位
2∣14	……… 1	
2∣7	……… 0	（反序）
2∣3	……… 1	
2∣1	……… 1	
0	……… 1	高位

小数部分

乘法	整数	
0.625 × 2 = 1.250	……… 1	高位
0.250 × 2 = 0.500	……… 0	（顺序）
0.500 × 2 = 1.000	……… 1	低位

即 $(59.625)_D=(111011.101)_B$，也可表示成111011.101B。

【例1-4】 将十进制数 $(427.34375)_D$ 转换成十六进制数。

解：

整数部分

		余数	
16 | 427			
16 | 26	……	11	↑ 低位
16 | 1	……	10	(反序)
0	……	1	高位

小数部分

		整数	
0.34375 × 16			
5.50000	……	5	高位
0.50000 × 16			(顺序)
8.00000	……	8	↓ 低位

结果是：$(427.34375)_D=(1AB.58)_{16}$，也可表示成1AB.58H。

1.3.3 二进制数与十六进制、八进制数间的转换

4位二进制数共有16种组合，而这16种组合恰好与十六进制数的16个数码相对应，故二进制数与十六进制数之间的转换可直接进行，即只要将二进制数的整数部分自右向左每4位分为一组，最后不足4位的用0补足；小数部分自左向右，也每4位分为一组，最后不足4位的在右边补0，再把每组二进制对应的十六进制数按原序写出即可。反之，要将十六进制数转换为二进制数，只需将每位十六进制数写成对应的4位二进制数后按原序写出即可。

【例1-5】 将二进制数 $(10110101011.100101)_B$ 转换成十六进制数。

解：因为 $10110101011.100101=\underline{0101}\ \underline{1010}\ \underline{1011}.\ \underline{1001}\ \underline{0100}$

↓ ↓ ↓ ↓ ↓

5 A B 9 4

所以 $(10110101011.100101)_B=(5AB.94)_H$

【例1-6】 将十六进制数 $(75E.C6)_H$ 转换成二进制数。

解：将每位十六进制数写成对应的4位二进制数：

$$(75E.C6)_H=(011101011110.11000110)_B=(11101011110.1100011)_B$$

从以上示例可以看出，同一个数用十六进制写出来的结果要比用二进制写出来的结果位数少得多，识别比较方便，再加上二进制与十六进制之间的相互转换非常方便，因此数字技术中也常采用十六进制。

八进制数转二进制数的规则是，将每位八进制数码分别用3位二进制数表示，并在这个0和1构成的序列去掉无用的前导0即得。

【例1-7】 将八进制数 $(5163)_O$ 转换成二进制数。

解：将每位八进制数码分别用3位二进制数表示，转换过程如下：

$$(5163)_O=(\underline{101}\ \underline{001}\ \underline{110}\ \underline{011})_2=(101001110011)_2$$

1.4 数字系统中数的表示方法和格式

在数字系统中主要使用二进制数，本节将介绍多种基于二进制数的编码表示方法。包

括用 BCD 码表示十进制数的方法；BCD 码和自然二进制码的区别；8421、2421 等 BCD 码、格雷码、余 3 码，及各种编码与二进制码的转换方法；ASCII 码、原码、反码与补码的基本概念及其表述。

1.4.1 十进制编码

数码不仅可以表示数量的大小，也可以表示不同的事物或不同的状态。当用于表示事物时，这些数码没有表示数量大小的含义。这种用数码或符号、文字来表示特定对象的过程称为编码，这些用于编码的数码称为代码。

在数字系统中，用 0 和 1 组成的二进制数码不仅可以表示数值的大小，还可以表示特定的信息。用 4 位二进制数组成一组代码来表示 0～9 这 10 个数字，这种代码称为二-十进制代码（Binary Coded Decimal），简称 BCD 码。表 1-3 给出 3 种常用的十进制编码。

表 1-3 3 种常用的十进制编码

十进制数	8421BCD 码	2421BCD 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

1. 8421BCD 码

8421BCD 码是最基本、最常用的一种编码方案，习惯上将其简称为 BCD 码。在这种编码方式中，每一位二进制代码都代表一个固定的数值，把每一位中的 1 所代表的十进制数加起来，得到的结果就是它所代表的十进制数码。由于代码中从左到右每一位中的 1 分别表示 8、4、2、1（权值），即从左到右，它的各位权值分别是 8、4、2、1。所以把这种代码叫做 8421 码。8421BCD 码是只取 4 位自然二进制代码的前 10 种组合。

2. 2421BCD 码

2421BCD 码是另一种有权码，也可以看作是 4 位二进制数。不过从左到右，它的各位权值分别是 2、4、2、1。与 8421 码一样，与每个代码等值的十进制数就是它表示的十进制数。例如与 2421 码 1110 等值的十进制数可按下式计算：

$$1\times2+1\times4+1\times2+0\times1=2+4+2+0=8$$

此外，在2421码中，0与9的代码、1与8的代码、2与7的代码、3与6的代码、4与5的代码均互为反码。

3. 余3码

余3码是一种特殊的BCD码，由8421BCD码加3后形成，所以叫做余3码。

1.4.2 格雷码

格雷码又称循环码，格雷码最重要的特点就是任意两个相邻的格雷代码之间，仅有一位不同，其余各位均相同。和二进制数相似，格雷码可以拥有任意的位数。表1-4中列出了4位格雷码及相应二进制码与十进制数的对照表。

表1-4 4位格雷码与二进制码对照表

十进制数	二进制码	格雷码	十进制数	二进制码	格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

格雷码与二进制码之间经常相互转换，具体方法如下。

(1) 二进制码到格雷码的转换：

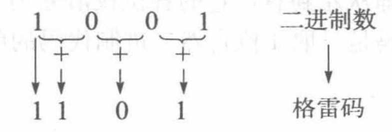
① 格雷码的最高位（最左边）与二进制码的最高位相同。

② 从左到右，逐一将二进制码的两个相邻位相加，作为格雷码的下一位（舍去进位）。

③ 格雷码和二进制码的位数始终相同。

【例1-8】把二进制数1001转换成格雷码。

解：把二进制数1001转换成格雷码的方法如下，转换后的格雷码是1101。



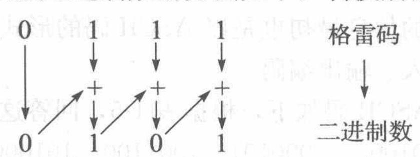
(2) 格雷码到二进制码的转换：

① 二进制码的最高位（最左边）与格雷码的最高位相同。

② 将产生的每个二进制码位加上下一相邻位置的格雷码位，作为二进制码的下一位（舍去进位）。

【例 1-9】 把格雷码 0111 转换成二进制数。

解：把格雷码 0111 转换成二进制数的方法如下，转换后的二进制数是 0101。



1.4.3 十进制数的 BCD 码表示方法

若用 BCD 码表示十进制数，只要把十进制数的每一位数码，分别用 BCD 码取代即可。反之，若要知道 BCD 码代表的十进制数，只要 BCD 码以小数点为起点向左、右每 4 位分成一组，再写出每一组代码代表的十进制数，并保持原排序即可。

【例 1-10】 求出十进制数 972.65_{10} 的 8421BCD 码。

解：将十进制数的每一位转换为其相应的 4 位 BCD 码。那么十进制数 972.65 就等于：

十进制	9	7	2	.	6	5
BCD	1001	0111	0010	.	0110	0101

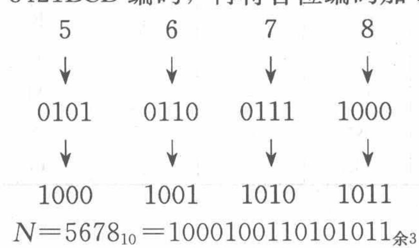
8421BCD 码：$\underline{1001}\ \underline{0111}\ \underline{0010}.\ \underline{0110}\ \underline{0101}_{8421BCD}$，即

$$972.65_{10} = 100101110010.01100101_{8421BCD}$$

注意：这是对一个十进制数进行编码或编号，是为了在某些情况下便于处理、表达或显示，或方便对编码后的数据用十六进制数来表达，而不是不同数制间的转换，因为它们不存在数值间的等价关系。

【例 1-11】 用余 3 码对十进制数 $N=5678_{10}$ 进行编码。

解：先对十进制数进行 8421BCD 编码，再将各位编码加 3 即可得到余 3 码。



所以有：
$$N=5678_{10}=1000100110101011_{余3}$$

1.4.4 字母数字码

计算机处理的数据不仅有数码，还有字母、标点符号、运算符号及其他特殊符号等。这些符号都必须使用二进制代码来表示，计算机才能直接处理。通常可同时用于表示字母和数字的编码称为字母数字码。

目前，在计算机和其他数字设备中广泛使用 ASCII 码，即美国信息交换标准码(American Standard Code for Information)。ASCII 码用 7 位二进制码来表示 128 个不同的数字、字母和符号，使用时加第 8 位作奇偶校验位。ASCII 码的编码如表 1-5 所示。

ASCII 码是一种常用的现代字母数字编码，用于计算机之间、计算机与打印机、键盘和视频显示等外部设备之间传输字符数字信息。

计算机用户从键盘输入的信息最初也是以 ASCII 码的形式进入计算机内部的。ASCII 码已成为微型计算机标准输入、输出编码。

【例 1-12】一组信息的 ASCII 码如下，根据表 1-5，回答这些信息是什么？

1001000　1000101　1001100　1010000

解：把每组 7 位码转换为等值的十六进制数，则有

48　45　4C　50

以此十六进制数为依据，查表 1-5 可确定其所表示的符号为

H E L P

表 1-5　美国信息交换标准码（ASCII 码）表

位765 / 位4321	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	]	l	|
1101	CR	GS	-	=	M	\	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

1.4.5　码制

此前所讨论的数都是正数，也就是说没有涉及数的符号问题。然而，数字系统既要处理正数，又要处理负数。那么，一个数的符号在数字系统中是如何表示的呢？带符号的数又如何在机器中表示呢？

在通常的算术运算中，用“＋”号表示正数，用“－”号表示负数，若用二进制数表达，则为“真值”，例如＋5 和－5 的真值分别是 101 和－101。但在数字电路中，正、负数的表示方法为：把一个数的最高位作为符号位，用“0”表示“＋”；用“1”表示“－”；连同符号位一起作为一个数，常用二进制数的表示方法有原码、反码和补码。

1. 原码表示法

用附加的符号位表示数的正负。符号位加在绝对值最高位之前（最左侧）。通常用符号位的“0”表示正数，用符号位的“1”表示负数，这种表示方法称为二进制的原码表示法。

例如十进制的＋37和－37的原码可分别写成：

十进制数　　＋37　　　　－37

二进制原码　0100101　　　1100101

（↑ 符号位　　　↑ 符号位）

小数＋53.625和－53.625的原码可分别写成：

十进制数　　＋53.625　　　－53.625

二进制原码　0110101.101　　11101010.101

（↑ 符号位　　　↑ 符号位）

因此，整数原码的定义为

$$[X]_{原码}=\begin{cases}X & 当\ 0\leqslant X<2^n\ 时\\ 2^n-X & 当\ -2^n<X\leqslant 0\ 时\end{cases}$$

2. 反码表示法

原码表示法虽然简单易懂，但在计算机中使用起来并不方便。如果进行两个异号数原码的加法运算，必须先判别两数的大小，然后从大数中减去小数，最后，还要判别结果的符号位，这样就增加了运算时间。事实上，在数字系统中更为适合的方法是采用补码表示法，而一个数的补码可以通过其反码获得。

反码的符号位表示法与原码相同，即符号“0”表示正数，“1”表示负数。与原码不同的是数值部分，即正数反码的数值和原码数值相同，而负数反码的数值是原码的数值按位求反。

【例 1-13】 用4位二进制数表示十进制数＋5和－5的反码。

解：可以先求十进制数所对应二进制数的原码，再将原码转换成反码。

十进制数　　＋5　　　　－5

二进制原码　0101　　　1101

二进制反码　0101　　　1010

（↑ 符号位　　　↑ 符号位）

即 $[+5]_{反}=0101$，$[-5]_{反}=1010$。

3. 补码表示法

在补码表示法中，正数的补码与原码和反码的表示相同。但是对于负数，从原码到补

码的规则是：符号位保持不变，数值部分则是按位求反，最低位加 1，或简称“求反加 1”。

(1) 整数补码的定义是

$$[X]_{补码}=\begin{cases}X & 当\ 0\leqslant X<2^n\ 时\\ 2^{n+1}+X & 当-2^n<X\leqslant 0\ 时\end{cases}$$

【例 1-14】用 4 位二进制数表示+5 和−5 的补码。

解：先求十进制数所对应二进制数的原码，再将原码转换成反码，然后将反码变为补码。

十进制数	+5	−5
二进制原码	0101	1101
二进制反码	0101	1010
二进制补码	0101	1010+1=1011
	↑ 符号位	↑ 符号位

即 $[+5]_{补}=0101$，$[-5]_{补}=1011$。

4 位有符号二进制数的原码、反码和补码表示法如表 1-6 所示。

表 1-6 4 位有符号数的表示

$b_3b_2b_1b_0$	原码	反码	补码	$b_3b_2b_1b_0$	原码	反码	补码
0111	+7	+7	+7	1000	−0	−7	−8
0110	+6	+6	+6	1001	−1	−6	−7
0101	+5	+5	+5	1010	−2	−5	−6
0100	+4	+4	+4	1011	−3	−4	−5
0011	+3	+3	+3	1100	−4	−3	−4
0010	+2	+2	+2	1101	−5	−2	−3
0001	+1	+1	+1	1110	−6	−1	−2
0000	+0	+0	+0	1111	−7	−0	−1

从表中可以看到，4 位二进制数有 16 种组合。原码和反码表示法能够表示数的范围都是−7 到+7；0 的表示不是唯一的，+0 和−0 有不同的编码。而在补码表示法中数的表示范围是−8 到+7，能够表示 16 个不同的数，0 的表示却是唯一的。

【例 1-15】求二进制数 $x=+1011$，$y=-1011$ 在 8 位存储器中的原码、反码和补码的表示形式。

解：无论是原码、反码和补码形式，8 位存储器的最高位为符号位，其他位则是数值部分的编码表示。在数值部分中，对于正数，原码、反码和补码按位相同，而对于负数，反码是原码的按位求反，补码则是原码的按位求反加 1。所以，二进制数 x 和 y 的原码、反码和补码分别表示如下：

$[x]_{原码}=00001011$　　$[x]_{反码}=00001011$　　$[x]_{补码}=00001011$

$[y]_{原码}=10001011$　　$[y]_{反码}=11110100$　　$[y]_{补码}=11110101$

【例 1-16】 求 $X=-1001010$ 的补码。

解：$[X]_{补}=2^8+(-1001010)=100000000-1001010=10110110$。

（2）定点小数（二进制小数）补码的定义是

$$[X]_{补}=\begin{cases}X & 当\ 0\leqslant X<1\ 时\\ 2+X & 当\ -1<X\leqslant 0\ 时\end{cases}$$

【例 1-17】 求 $X_1=+0.1011011$ 和 $X_2=-0.1011011$ 的补码。

解：$[X_1]_{补}=0.1011011$

$[X_2]_{补}=2+(-0.1011011)=10-0.1011011=1.0100101$

1.4.6 用补码进行二进制数运算

以上介绍了二进制数的 3 种表示方法，由于它们的形成规则不同，因此运算方法也各不相同。以下介绍的内容表明，使用补码进行二进制数的运算具有很大的优势。

1. 原码运算

原码中的符号位不参加运算。同符号数相加作加法；不同符号数相加作减法。显然，原码的运算法则与传统的加减法相同，由于在运算中要对符号进行判断后才能决定加减算法，这在使用二进制运算的计算机中是十分不方便的。

2. 补码运算

运算时，符号位和数值一起参加运算，不单独处理，这是补码运算的最大优势。

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}\quad [X-Y]_{补}=[X]_{补}+[-Y]_{补}。$$

3. 反码运算

运算时符号位与数值一起参加运算，如果符号位产生了进位，则此进位应加到和数的最低位，称为循环进位。

$$[X+Y]_{反}=[X]_{反}+[Y]_{反}\quad [X-Y]_{反}=[X]_{反}+[-Y]_{反}$$

【例 1-18】 设 $X=+1011101$，$Y=+0011010$，求 $Z=X-Y$。

解：(1) 原码运算。$[X]_{原}=01011101$，$[Y]_{原}=00011010$

因为 $|X|>|Y|$，所以 X 作被减数，Y 作减数，差值为正。

$$\begin{array}{r}0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\\ -\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\\ \hline 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\end{array}$$

即 $[Z]_{原}=01000011$，其真值为 $Z=+1000011$。

(2) 反码运算：$[X]_{反}=01011101$，$[-Y]_{反}=11100101$

运算中，符号位产生了进位，因此需将此进位加到和的最低位。从运算过程可见，反码加法运算后，须判断是否需要作循环进位运算，而循环进位运算又相当于一次加法运算，因此会影响运算器的运算速度。

```
      0  1  0  1  1  1  0  1
 +    1  1  1  0  0  1  0  1
 ---------------------------
 (1)  0  1  0  0  0  0  1  0
 +└──────────────────────→ 1
 ---------------------------
      0  1  0  0  0  0  1  1
```

即 $[Z]_{反}=01000011$，其真值为 $Z=+1000011$。

（3）补码运算：$[X]_{补}=01011101$，$[-Y]_{补}=11100110$

```
        0  1  0  1  1  1  0  1
 +      1  1  1  0  0  1  1  0
 -----------------------------
   (1)  0  1  0  0  0  0  1  1
    │
舍弃←┘
```

即 $[Z]_{补}=01000011$，其真值为 $Z=+1000011$。

补码运算时，符号位和数值位一起参加运算，若符号位产生进位，则将进位位丢弃，不用作循环进位运算。由于不需要作进位判别，从而简化了电路设计，给运算带来方便。

【例 1-19】用 4 位二进制数的补码形式运算 7－5 和 4－6。

解：$7-5=7+(-5)=[7]_{补}+[-5]_{补}=0111+1011=0010$（丢弃进位）$=2$

$4-6=[4]_{补}+[-6]_{补}=[0100]_{补}+[1110]_{补}=0100+1010=[1110]_{补}$

$=[1010]_{原码}=-2$

习　　题

1-1　什么是数字信号？什么是模拟信号？试举例说明。

1-2　将下列二进制数转换成十进制数和十六进制数：

（1）1011001　　（2）0.101011　　（3）1011.0101　　（4）11.01101

1-3　十进制数 $(78.25)_{10}$ 转换成十六进制数是（　　），转换成二进制数是（　　），转换成八进制数是（　　），转换成 8421BCD 码为（　　）。

1-4　一组 8 位数字信号的波形如图 1-1 所示，试求该波形所代表的二进制数。

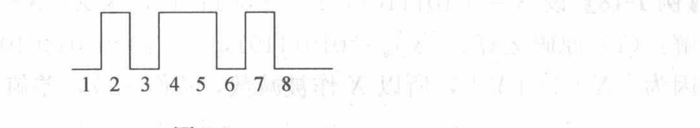


图 1-1

1-5　将下列数值转换成十进制数：

（1）$(1010.1011)_2$　（2）$(101011)_2$　（3）$(110.1011)_2$

（4）$(2E5.3)_{16}$　（5）$(35.26)_8$

1-6　把下列 4 个不同数制的数按从大到小的次序排列：

（1）$(376.125)_D$，$(110000)_B$，$(17A)_H$，$(67)_O$

（2）$(76.125)_D$，$(27A)_H$，$(10110)_B$，$(56)_O$

1-7 将十进制数（43）和（14.625）分别转换为二进制数、八进制数和十六进制数。

1-8 将下列十进制数转换为十六进制数：

（1）500D （2）59D （3）0.34D （4）1002.45D

1-9 将十六进制数 23F.45H 和 A040.51H 转换为二进制数。

1-10 将十六进制数 103.2H 和 A45D.0BCH 转换为十进制数。

1-11 写出下列十进制数的 8421BCD 码和余 3 码：

（1）675 （2）9536

1-12 已知格雷码为 1000，求对应的二进制码；已知十进制数为 92，求对应的余 3 码。

1-13 将下列十进制数转换为二、八、十六进制数和 8421BCD 码（要求转换误差不大于 2^{-4}）：

（1）43 （2）127 （3）254.25 （4）2.718

1-14 将二进制数 1100110 转换成余 3 码为（ ），转换成格雷码为（ ）。

1-15 设真值 $X=-0101$，则 X 的原码为（ ），反码为（ ），补码为（ ）。

1-16 写出下列各数的原码、反码和补码：

（1）+1011011 （2）−1010110 （3）+0.1011101 （4）−0.1101001

1-17 若 $X=+1010101$，$Y=+1101101$，求 $[X-Y]_{补}$。

1-18 两个带符号的二进制的真值 $N_1=-01010$，$N_2=+10011$，求它们的原码、反码和补码。

1-19 将下列有符号的十进制数转换成相应的二进制数真值、原码、反码和补码：

（1）+109 （2）−37 （3）+124 （4）−29

1-20 将有符号的十进制数+122 转换成相应的二进制数真值、原码、反码和补码。

1-21 求 $[X]_{原}=1.1101$ 的真值和补码；求 $[X]_{反}=0.1111$ 的补码。

1-22 将十进制数+115 和−38 转换成相应的二进制数真值、原码、反码和补码。

1-23 设 $X=+1110101$，$Y=+0101101$，分别用它们的原码、反码、补码，计算 $Z=X-Y$。

1-24 用 4 位二进制数的补码形式分别计算 2−9、6+7、8−2、4−9。给出计算过程。

第2章 逻辑门功能及其电路特性

逻辑门是完成二值（1和0）逻辑运算或二进制数据运算的基本元件。这些元件可以由逻辑门电路来实现。基本逻辑门电路，即门电路，是组成数字电路和数字系统的基本单元。门电路中的半导体器件一般工作在开关状态。本章将系统地介绍这些基本逻辑门的电路及其电路特性，以及基于逻辑代数的逻辑运算与图形描述符号。此外，针对实际应用，介绍三态逻辑门和集电极开路输出门的电路特性，以及介绍 TTL 集成门和 CMOS 集成门的逻辑功能、特性及性能参数。

2.1 基本逻辑门

实现基本逻辑运算和二进制数据运算的单元电路称为逻辑门电路，逻辑门电路的功能可以由多种方式来表述，如逻辑代数、表格或图形等。在逻辑代数中，最基本的逻辑运算有与、或、非3种。每种逻辑运算代表一种函数关系，此函数关系可用逻辑符号写成逻辑表达式来描述，也可用文字来描述，还可用表格或图形来描述。

2.1.1 逻辑代数的3种基本运算模型

如上所述，最基本的逻辑运算有与、或、非3种，这3种逻辑运算可以形象地用电灯与开关的串并联电路来描述，它们分别如图2-1所示。

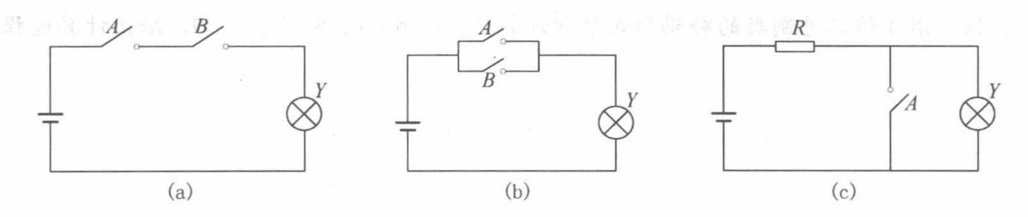


图2-1　与、或、非逻辑说明示例

图2-1（a）电路中，仅当在两个开关 A 和 B 都接通时灯才会亮。也就是说，导致灯亮结果的所有条件同时具备时，结果才会发生。这种因果关系被称为“逻辑与”，或者“与”，可以用点“·”、文字“AND”、符号“&”、或特定的图形符号来表示。如 $Y=A\cdot B$、或者 $Y=A \& B$ 等。亮灯与否的结果，可以用1表示灯亮，0表示灯闭。

图2-1（b）电路中，仅当两个开关 A 和 B 中任何一个开关接通后，灯才会亮。也就

是说，只要导致亮灯结果的条件中有任何一个条件具备时，结果就会发生。这种因果关系被称为“逻辑或”，或者“或”，可以用加号“+”、文字“OR”、符号“|”，或特定的图形符号来表示，如$Y=A+B$，或者$Y=A \mid B$等。

在图 2-1（c）电路中，开关接通时灯不亮，而当开关断开时灯反而会亮。这就是说，当某条件具备时，结果不会发生；而当某条件不具备时，结果必定发生。这种因果关系被称为“逻辑非”，或者“非”，这可以在逻辑变量上用横杠，即取非号、或文字“NOT”、符号“!”，或特定的图形符号等来表示，如$Y=\overline{A}$，或$Y=!A$等。

由此便可得到图 2-1 所对应的逻辑功能表，表 2-1 是对应图 2-1（a）的与逻辑功能表；表 2-2 是对应图 2-1（b）的或逻辑功能表；而表 2-3 则是对应图 2-1（c）的非逻辑功能表。

表 2-1　与逻辑功能表

开关 A	开关 B	灯 Y
断开	断开	灭
断开	闭合	灭
闭合	断开	灭
闭合	闭合	亮

表 2-2　或逻辑功能表

开关 A	开关 B	灯 Y
断开	断开	灭
断开	闭合	亮
闭合	断开	亮
闭合	闭合	亮

表 2-3　非逻辑功能表

开关 A	灯 Y
断开	亮
闭合	灭

若用A、B两个逻辑变量表示开关的状态，并规定 1 表示开关接通，0 表示开关断开；用Y表示灯的状态，1 表示灯亮，0 表示灯灭。这样就可以根据表 2-1、表 2-2 和表 2-3 列出用 0、1 表示的与、或、非的逻辑运算（逻辑操作）的表格，这种表格称为逻辑真值表（简称真值表），它将输入变量（如A和B）所有可能的取值组合与其对应的输出变量（如Y）的值逐个列举出来。

逻辑真值表是描述逻辑功能的一种重要方法，后面的介绍中将经常用到。

表 2-1、表 2-2 和表 2-3 所对应的逻辑真值表，分别对应于表 2-4 的与逻辑真值表、表 2-5 的或逻辑真值表和表 2-6 的非逻辑真值表。

表 2-4　与逻辑真值表

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

表 2-5　或逻辑真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

表 2-6　非逻辑真值表

A	Y
0	1
1	0

2.1.2 基本逻辑符号

在逻辑代数中，与运算符号用“·”表示，或运算符号用“+”表示，非运算符号用“—”表示。

当A和B作与运算，结果为Y时，逻辑代数，或布尔函数（Boolean Function）可写作$Y=A\cdot B$；当A和B作或运算，结果为Y时，逻辑代数写作$Y=A+B$；当A作非运

算，结果为Y时，逻辑代数可写作$Y=\overline{A}$。

利用这些运算符号可以将表2-4、表2-5和表2-6逻辑真值表改写为如下的与、或、非运算规律：

与运算	或运算	非运算
$0\cdot 0=0$	$0+0=0$	$\overline{0}=1$
$0\cdot 1=0$	$0+1=1$	$\overline{1}=0$
$1\cdot 0=0$	$1+0=1$	
$1\cdot 1=1$	$1+1=1$	

由这些运算规律可推出一般形式的运算规则（更详细的内容将在第3章中介绍）：

与运算	或运算	非运算
$A\cdot 0=0$	$A+0=A$	$\overline{\overline{A}}=A$
$A\cdot 1=A$	$A+1=1$	$\overline{A}+A=1$
$A\cdot A=A$	$A+A=A$	$\overline{A}\cdot A=0$

能够实现“与”、“或”、“非”逻辑运算功能的电路分别称为“与门”、“或门”和“非门”。门电路的逻辑功能还可以用图形方式来表示。美国国家标准学会（ANSI）和电气与电子工程师协会（IEEE）于1984年制定了关于二进制逻辑图形符号的标准，即ANSI/IEEE 1984标准，1991年对该标准做了补充和修订，推出了ANSI/IEEE 1991标准。补充和修订后的标准既允许使用原ANSI/IEEE 1984版的矩形轮廓图形符号，也允许使用ANSI/IEEE 1991版的具有特定外型的图形符号。

由于目前国际上流行的大多数数字技术教材、数字系统设计技术资料和主流EDA软件中，一直流行采用ANSI/IEEE 1991版的图形符号，因此本教材根据这一流行趋势，主要使用此类图形符号。图2-2所示即ANSI/IEEE 1984版（图2-2（a））与ANSI/IEEE 1991版（图2-2（b））的国际标准部分逻辑门符号，以及对应的逻辑表式（图2-2（c））。

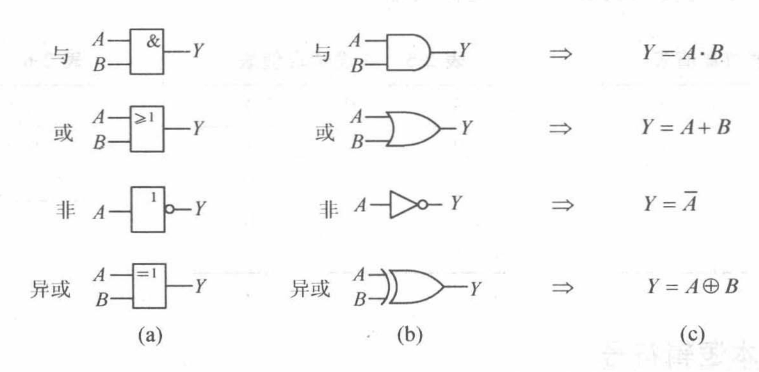


图2-2 两种IEEE标准基本逻辑门符号的比较

图2-2中的逻辑门符号最多只有两个逻辑信号输入端，对于多个输入端，也有类似图形和逻辑表述。图2-3和图2-4所示的逻辑符号分别表示的是3输入与门和8输入与门、3输入或门和8输入或门。

逻辑图形符号在逻辑运算中可以看成是一个抽象的逻辑符号，但也可以看成是一个电



图 2-3 3 输入与门和 8 输入与门　　　图 2-4 3 输入或门和 8 输入或门

路元件符号。因为实际的逻辑门就是由数字电路构成的，所以也可以通过观察逻辑门电路的输入输出电压信号来了解其逻辑功能，或称对逻辑电路的信号响应的考察。

如图 2-5（a）所示，A 和 B 的信号是向 2 输入与门输入的电压信号波形；图 2-5（b）是 2 输入与门逻辑符号；图 2-5（c）是对应于输入波形的输出波形 Y。图 2-5（c）显示，当输入波形 A 和 B 同时为高电平时对应逻辑 1，而在其他情况下 Y 输出为低电平（对应逻辑 0）。

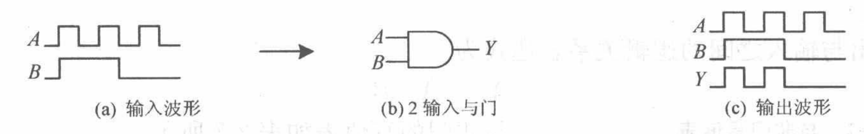


图 2-5 2 输入与门及其输入和输出波形

图 2-6 所示的是针对 2 输入或门的输入、输出电压信号波形的情况。

对于图 2-6（a）所示的输入波形，经过 2 输入或门（图 2-6（b））后的输出波形 Y 如图 2-6（c）所示。由图可见，当输入波形 A 和 B 之一、或全部为高电平时，Y 的输出波形为高电平；只有当输入波形 A 和 B 皆为低电平时，输出信号 Y 才为低电平（对应逻辑 0）。

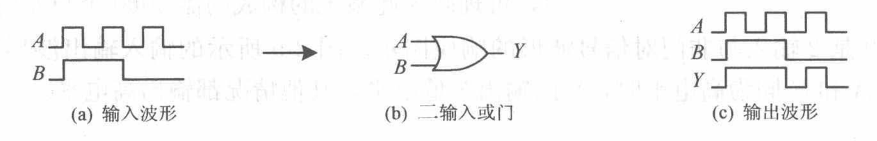


图 2-6 2 输入或门及其输入和输出波形

图 2-7（a）所示的是向非门（或称反相器）输入的信号波形。图 2-7（c）给出了非门的输出波形。由图 2-7 可见，当输入波形为高电平时，输出就为低电平；反之亦然。

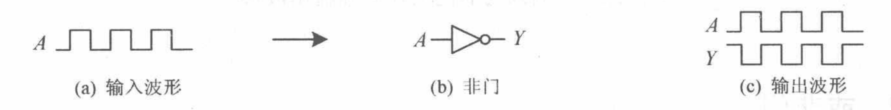


图 2-7 非门及其输入和输出波形

2.2 复合逻辑门

本节介绍与非、或非、异或、同或的复合逻辑运算，及其相应的逻辑门，即与非门、或非门、异或门、同或门的逻辑功能、真值表描述及信号波形响应情况。

基本逻辑运算的复合叫做复合逻辑运算。而实现复合逻辑运算的电路称为复合逻辑

门。最常用的复合逻辑门有与非门、或非门、与或非门和异或门等。

2.2.1 与非门

与运算后再进行非运算的复合运算称为“与非”运算。实现与非运算的逻辑电路称为“与非门”。一个与非门有两个或两个以上的输入端和一个输出端。2 输入端与非门的逻辑符号如图 2-8 所示。

(a) 与门和非门组合　　(b) 与非门

图 2-8　2 输入与非门的图形符号

其输出与输入之间的逻辑关系表达式为

$$Y=\overline{A\cdot B}$$

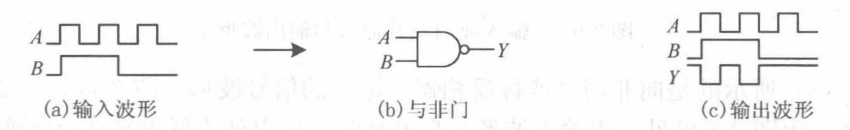
与非门的真值表如表 2-7 所示。

表 2-7　与非门真值表

A	B	$Y=\overline{A\cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

从理论上看，在数字系统中可以使用与非门来实现任何逻辑功能的逻辑电路。因此，可以将与非门看成是一种具有逻辑完备性的通用逻辑门，于是，常用它作为衡量一个数字系统占用的逻辑资源的多少及系统规模大小的最小单元。例如说某一数字系统的逻辑规模有 5000 门，可理解为此系统的构成约需 5000 个与非门。

图 2-9 是 2 输入与非门对信号波形的响应图示。图 2-9 所示的输入输出波形表示，当输入波形 A 和 B 同为高电平时，Y 的输出为低电平，其他情况都输出高电平。



(a) 输入波形　　(b) 与非门　　(c) 输出波形

图 2-9　2 输入与非门的输入和输出波形

2.2.2 或非门

或运算后再进行非运算的复合运算称为“或非”运算。实现或非运算的逻辑电路称为“或非门”。一个或非门有两个或两个以上的输入端和一个输出端。2 输入端或非门的逻辑符号如图 2-10 所示。

其输出与输入之间的逻辑关系表达式为

$$Y=\overline{A+B}$$

或非门的真值表如表 2-8 所示。

(a)或门和非门组合 (b)或非门

图 2-10 2 输入或非门的逻辑符号

表 2-8 或非门真值表

A	B	$Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

和与非门一样，或非门也可用来实现任何逻辑功能的逻辑电路。因此，或非门也是一种通用逻辑门。

图 2-11（a）所示的是向 2 输入或非门输入的信号波形，其输出波形 Y 如图 2-11（c）所示：当输入波形 A、B 中有一个或均为高电平时，Y 的输出波形就为低电平，其他情况都输出高电平。

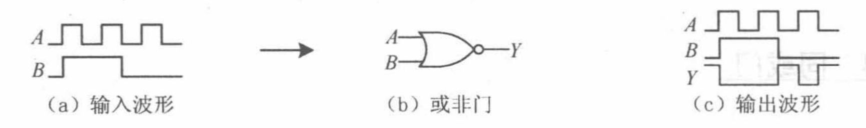


图 2-11 或非门的输入和输出波形

2.2.3 异或门

在集成逻辑门中，异或逻辑主要为 2 输入变量的门，对 3 输入或更多输入变量的逻辑，都可由 2 输入门导出。所以，常见的异或逻辑是 2 输入变量的情况。

2 输入变量的异或逻辑的表述是，当两个输入端取值不同时，输出为 1；当两个输入端取值相同时，输出为 0。

实现异或逻辑运算的逻辑电路称为“异或门”。如图 2-12 所示为 2 输入异或门的逻辑符号，相应的逻辑表达式为

$$Y=A\oplus B=\overline{A}\cdot B+\overline{B}\cdot A$$

或可表为

$$Y=A\oplus B=\overline{A}B+\overline{B}A$$

其真值表如表 2-9 所示。

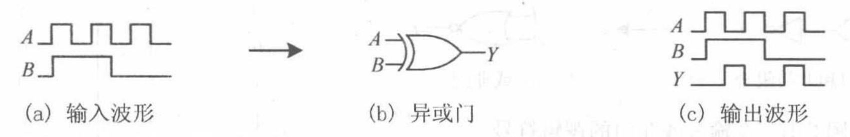
图 2-12 异或门逻辑符号

表 2-9 异或门真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

图 2-13（a）所示的是向异或门输入的信号波形，获得图 2-13（c）所示的输出波形

Y。图 2-13（c）表明，当输入波形 A 和 B 有且仅有一个为高电平时的时间段内，Y 的输出波形就为高电平。



(a) 输入波形　　(b) 异或门　　(c) 输出波形

图 2-13　异或门的输入和输出波形

对于多变量的异或逻辑运算，常以两变量的异或逻辑运算的定义为依据，来对输出进行推证。N 个变量的异或逻辑运算输出值和输入变量取值的对应关系是：输入变量的取值组合中，有奇数个 1 时，异或逻辑运算的输出值为 1；反之，输出值为 0。

异或逻辑运算有许多实际应用，如对逻辑数据的加密，或通过可编程逻辑电路十分方便地对逻辑信号取反等。

2.2.4　同或门

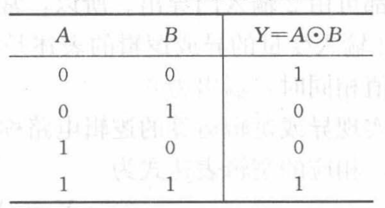
同或逻辑即异或运算之后再进行非运算，则称为“同或”运算。实现同或运算的电路称为“同或门”。同或门的逻辑符号如图 2-14 所示。

2 输入同或运算的逻辑表达式为

$$Y = A \odot B = \overline{A}\,\overline{B} + AB$$

同或逻辑的真值表如表 2-10 所示。

表 2-10　同或门真值表



A	B	$Y=A\odot B$
0	0	1
0	1	0
1	0	0
1	1	1

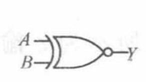


图 2-14　同或门的逻辑符号

如图 2-15（a）所示的是向同或门输入的信号波形，其输出波形 Y 如图 2-15（c）所示。当输入波形 A 和 B 有且仅有一个为高电平时的时间段内，输出波形 Y 就为低电平。

与多变量的异或逻辑运算相同，多变量的同或逻辑运算也常以两变量的同或逻辑运算的定义为依据进行推证。N 个变量的同或逻辑运算的输出值和输入变量取值的对应关系是：输入变量的取值组合中，有偶数个 1 时，同或逻辑运算的输出值为 1，反之为 0。

(a)输入波形　　(b)同或门　　(c)输出波形

图 2-15　同或门的输入和输出波形

2.3 其他辅助门电路

本节中将介绍三态逻辑门，或称三态门的逻辑功能、集电极开路输出逻辑门的逻辑功能，以及相关电路的分析和应用说明。

2.3.1 三态门

三态门是传输门的一种，主要用于对信号传输的控制。三态门（简称 TS 门）除了有此前已介绍的高电平和低电平（即逻辑 1 和逻辑 0）两种逻辑状态或逻辑值外，还拥有第 3 种逻辑状态，即高阻状态（通常记为 Z），也称为禁止状态，或电路断开状态。在第 3 种状态下，三态门的输出端相当于悬空（电路断开），此时的输出端就好像一根空头的导线，其电压值可浮动在高低电平之间的任意数值上。

三态门的构成是在普通逻辑门电路的基础上增加一些专门的控制电路，以及一个控制输入端，即三态使能端：EN（Enable）端。通过 1 和 0 逻辑电平可控制此端。图 2-16 给出了三态门的逻辑符号。

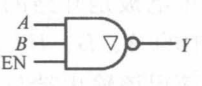
图 2-16（a）为控制端（EN）高电平有效型三态门，它的逻辑功能可表达为：当 EN=1 时（EN 输入为高电平时），$Y=A$，即 Y 直接输出来自 A 的信号；而当 EN=0 时，Y 呈高阻态，即等同于断开状态，可表述为：$Y=Z$。

(a)高电平使用　(b)低电平使用

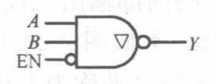
图 2-16　三态门

图 2-16（b）所示的三态门逻辑符号则正好相反，其控制端为低电平有效型三态门。其逻辑功能可表达为：当 EN=0 时（EN 输入为低电平时），三态门工作，即 $Y=A$，而当 EN=1 时，$Y=Z$。

图 2-17 是三态使能控制的与非门，其真值表如表 2-11 所示。当 EN=1 时，三态门工作，实现正常"与非"功能，即输出 $Y=\overline{A\cdot B}$；当 EN=0 时，三态门禁止，输出 $Y=Z$，即输出端 Y 呈现高阻态。表中的"x"表示任意电平，1 或 0。



(a)控制端高电平有效



(b)控制端低电平有效

图 2-17　三态与非门的逻辑符号

表 2-11　图 2-17（a）的三态门真值表

使能端	数据		输出端
EN	A	B	Y
0	x	x	高阻态 Z
1	$Y=\overline{A\cdot B}$		

同样，图 2-17（b）在 EN 控制端上加了一个小圆圈，表示低电平有效。

当三态门输出端处于高阻状态时，该门电路表面上（物理结构上）仍与整个电路系统相连接，但实际上对整个系统的逻辑功能和电气特性均不发生任何影响，如同没把它接入系统一样。三态门是数字系统在采用总线结构时，对接口电路提出的要求。因此，三态门在总线接口电路中得到了广泛的应用。

总线是一个对来自不同信号源，能分时传输这些不同来源信号或数据的单通道信号传

输系统。

图 2-18 所示的电路结构就是利用三态门功能，来实现多路数据在总线上的分时传送目的的。为实现这一功能，只要适当控制各个门的 EN_i 输入端，轮流定时地使各个 EN_i 端为 0（输出有效），并且在任何时刻只有一个 EN_i 端为 0，这样就可以把各个门的输出信号轮流传送到总线上，而不会发生数据传输错误。

显然，必须保证在任何时刻只有一个三态门被选通。只有一个门向总线传送数据是十分重要的，否则就会造成总线上的数据混乱（数据冲突），并且有可能损坏处于导通状态的半导体输出器件。

传送到总线上的数据可以同时被多个负载门接收，也可以在控制信号作用下，让指定的负载门接收。

利用三态门还可以实现数据的双向传输，这有许多实际的应用，如存储器的数据写入或读出。图 2-19 所示的电路结构就是一个利用两个三态门构建的双向传输门。

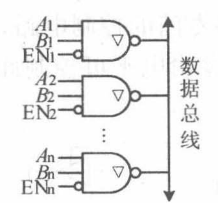


图 2-18　三态门用于总线传输图

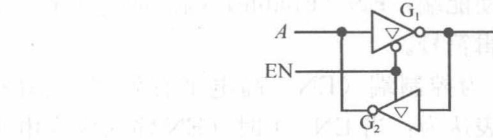


图 2-19　用三态门实现数据双向传输

图 2-19 中，门 G_1 和门 G_2 为三态反相器，即拥有三态控制输出的非门。门 G_1 低电平控制有效，门 G_2 高电平控制有效。当三态使能端 EN=0 时，门 G_1 选通，门 G_2 禁止，数据可以从 A 传到 B；当三态使能端 EN=1 时，门 G_2 选通，门 G_1 禁止，数据可以从 B 传到 A，不过结果必须取反。

2.3.2　集电极开路门

集电极开路门，简称 OC 门，其特点是门电路内部输出三极管的集电极是开路的。三极管是数字电路中的基本元件，通常有三个引脚，类似图 2-20（a）中的 A、B、F。在使用时，必须外接上拉电阻 R_P（将电阻的一端接于高电平称为上拉），使得该输出端与直流电源相连。多个 OC 门输出端相连时，可以共用一个上拉电阻 R_P。

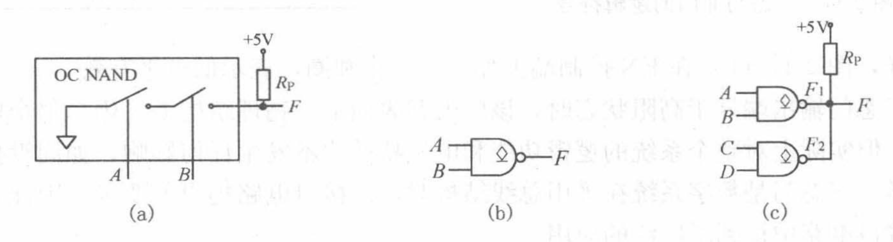


图 2-20　OC 与非门的开关级描述及其线与电路

为了能暂时不必介绍三极管的工作原理，图 2-20（a）给出了一个 OC 与非门的开关级等效电路结构图，其逻辑符号如图 2-20（b）所示。它的逻辑功能是，只有当 A 和 B 都为 1 时（A、B 开关都连通时），输出 F 才为 0；否则，输出 F 与逻辑门脱离了连接，呈现高阻态（这里假设没有连接上拉电阻 R_P）。这时如果希望使此电路实现与非门逻辑特性：$F=\overline{A \cdot B}$，则须用上拉电阻将它拉至逻辑 1。对于 5V 工作电压的逻辑器件，应该如图 2-20（a）那样使电阻 R_P 接在工作电压＋5V 上，即上拉（如果接 0V 称为下拉）。电阻阻值一般在 1～10kΩ 之间。图 2-20（c）是 OC 与非门的应用示例。

为了帮助读者了解 OC 门的一些实用电气特性，以下介绍几则应用实例。

1. 实现线与功能

由于 OC 门具有三态输出特性，若两个 OC 与非门输出端作并联连接，其电路可以如图 2-20（c）所示的那样连接，并联后实现的逻辑功能对应的布尔函数表式应该是

$$F = F_1 \cdot F_2 = \overline{AB} \cdot \overline{CD}$$

显然，F 与 F_1、F_2 之间为与逻辑关系（条件是必须有一个上拉电阻）。由于这种与逻辑是两个 OC 门的输出线直接相连实现的，故称作“线与”。

同理，图 2-21 中的 4 个 OC 与门线与的输出逻辑表达式如下：

$$Y = A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H$$

可以看出，4 个 2 输入 OC 与门的线与连接构成了一个 8 输入的与门。

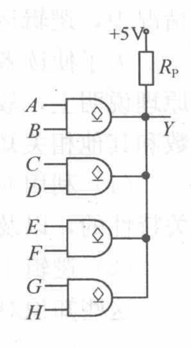


图 2-21 4 个 OC 与门

虽然利用集电极开路门可以使门的输出端并联起来，获得附加的逻辑功能，但是，由于上拉电阻 R_P 的应用而限制了门电路的工作速度，或开关速率。这是因为电阻 R_P 越大，电路的输出阻抗越大，暂态响应的时间因子也越大。因此，OC 逻辑门电路在现代数字系统设计中已极少用到了，它仅在低速接口电路中有一定的实用价值。

通常，OC 门和三态门都可以允许输出端直接并接在一起，用来实现多路信号在总线上的分时传送。但是三态门在使用时不需要再另外加接上拉电阻，所以更经济、更高速，因此在现代逻辑设计中，三态门已经完全取代了 OC 门。

2. 实现电平转换

如图 2-20 所示，当线与的 OC 门 F_1、F_2 的输出级都截止时（呈高阻态），由于上拉电阻的作用，F 输出高电平，这个高电平就等于其上拉的电源电压 V_{CC}（上拉电阻所接的电压：$V_{CC}=+5V$）。这个 V_{CC} 的电平值可以不同于门电路本身的电源，所以只要根据要求来选择 V_{CC}，就可以得到 F 输出所需的不同的电平值。

在某些情况下，数字系统在其接口部分（与外部设备相连接的地方）需要转换输出电平，且对信号速度没有很高要求的情况下，常使用此类逻辑门来完成电平的转换。如图 2-22 所示，把上拉电阻接到 $V_{CC}=10V$ 的电源上，这样在 OC 门就可以实现在输入端接受普通的 TTL 电平（高电平定义为 3.0～5V 间的电压值），而输出端可提供达 10V 的高电平。

因而输出可适应于需要较高电平的器件，如荧光数码管、MOS译码器等。

3. 用作驱动器

也可用OC门来直接驱动发光二极管、指示灯、继电器或脉冲变压器等需要大电流的器件。图2-23是用来驱动发光二极管的电路。当OC门输出低电平时，电流通过上拉电阻R_P，经过发光二极管流入OC门的地，于是发光二极管导通发光；当OC门输出高电平时，电流通路被断开，发光二极管截止。考虑到驱动的需要，此电路中的上拉电阻R_P的阻值要视发光二极管的特性选取，通常小于1kΩ。



图2-22 实现电平转换　　图2-23 驱动发光二极管

2.4 集成电路逻辑门

在2.1节中曾利用电灯与开关的串并联电路来形象地说明逻辑运算的关系，然而实际情况中，逻辑运算是在数字电路中通过半导体器件的开关特性实现的。

为了使读者能更加深入地了解实现逻辑功能和逻辑运算的电路原理，本节将重点放在原理说明上，故以下简要介绍基于CMOS及TTL集成逻辑门的概念、工作原理、性能参数和其他相关知识。本节的学习重点有两个：

(1) 利用MOS晶体管的等效开关模型，了解集成电路中的半导体元件是如何实现开关特性的，以及基于这些开关特性是如何实现逻辑功能和逻辑运算的。

(2) 逻辑1和0，或者说二进制数的1和0，在实际电路中是如何表达的。

这些知识对于后续的实验和设计活动十分重要。

2.4.1 逻辑门及其基本结构与工作原理

把若干个有源器件和无源器件及其连线，按照一定的功能要求，制作在一块半导体基片上，这样的产品叫集成电路。若它完成的功能是逻辑功能或数字功能，则称为数字集成电路。最简单的数字集成电路是集成逻辑门。

集成电路比分立元件电路（如三极管）有许多显著的优点，如体积小、耗电省、重量轻、可靠性高等。所以集成电路一出现就受到人们的极大重视并迅速得到广泛的应用。习惯上，数字集成电路的规模一般是根据门的数目来划分的。

比较早期的划分方法是，小规模集成电路（SSI）约为10个门，中规模集成电路（MSI）约为100个门，大规模集成电路（LSI）约为1万个门，而超大规模集成电路（VLSI）则在100万门以上。

显然这种划分与归类方法在今天已经过时了，因为现在的集成电路规模概念上，小规

模至少也应该有上千个逻辑门。

现在，集成电路的集成规模已有了巨大的进步。系统的等效门规模达千万门的逻辑器件已是平常事，几十个门的小规模逻辑器件早已退出了应用与设计领域。因此器件的集成规模的划分也会不断发生变化。

在逻辑电路中，通常用晶体管来实现开关的功能。对于传统的集成电路逻辑门，按照其组成的不同，可分为两大类，一类是双极型晶体管逻辑门；另一类是单极型的绝缘栅场效应管逻辑门。

双极型晶体管逻辑门主要有 TTL 门（Transistor-Transistor Logic，即晶体管-晶体管逻辑电路）、ECL 门（射极耦合逻辑门）和 I^2L 门（集成注入逻辑门）等。

目前最流行的能够实现开关功能的晶体管是金属氧化物半导体场效应晶体管（MOSFET）。这就是单极型 MOS 门，它主要有 PMOS 门（P 沟道增强型 MOS 管构成的逻辑门）、NMOS 门（N 沟道增强型 MOS 管构成的逻辑门）和 CMOS 门（利用 PMOS 管和 NMOS 管构成的互补电路构成的门电路，故又叫互补 MOS 门）。

CMOS 是 Complementary Symmetry Metal Oxide Semiconductor（互补对称金属氧化物半导体）的简称。在数字集成电路中，CMOS 集成电路的应用最为广泛。

N 沟道的 MOS 晶体管，即 NMOS 晶体管，其图形符号如图 2-24（a）所示。NMOS 晶体管有 4 个接线端，分别是源极 S（Source）、栅极 G（Gate）、漏极 D（Drain）和衬底 B。

图 2-24（b）是 NMOS 晶体管的两种简化符号。NMOS 管的简化符号表明，NMOS 晶体管有 3 个接线点（3 个电极），即源极 S、栅极 G 和漏极 D。栅极通常是一层金属，或者多晶硅；源极和通道之间有一层很薄的二氧化硅绝缘层。当栅极 G 和源极 S 之间加上正向电压 V_{gs}（S 可接地），则栅极 G 和沟道之间的电场穿透氧化物，在通道中产生一个反转沟道，这个沟道和源极漏极的特性一致。因此形成一层导电层，使漏极 D 和源极 S 之间产生电流（产生导通通道）。因此改变栅极和源极之间的电压，可以改变该沟道的导电特性，从而改变漏极和源极之间电流的大小。否则，漏极 D 和源极 S 之间将等效于电路断开，即所谓截止。因此可通过控制“开关”G，来控制 D 与 S 间通道的接通与否。不过，栅极和源极、漏极之间并没有直接联系的通道，只有电容耦合，这在高速电路中会产生一定功耗。

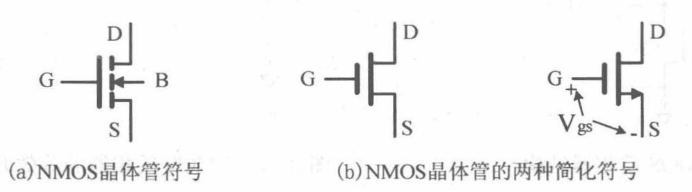


图 2-24 NMOS 晶体管的图形符号

P 沟道的 MOS 晶体管，即 PMOS 晶体管，其图形符号如图 2-25（a）所示。图中也有 4 个接线端，分别是源极 S（Source）、栅极 G（Gate）、漏极 D（Drain）和衬底 B；图 2-25（b）是PMOS 晶体管的两种简化符号。PMOS 的开关控制特性与 NMOS 正好相反。如图 2-25（b）所示，当 PMOS 管的 G 和 S 间加一反向电压 V_{gs}，则 D 与 S 间将产生一个电路通道。即可通过控制“开关”G，来控制 D 与 S 间通道的接通与否。

(a) PMOS晶体管符号 (b) PMOS晶体管的两种简化符号

图 2-25 PMOS晶体管的图形符号

CMOS（Complementary MOS）电路，即互补MOS。NMOS和PMOS用互补的方式联合使用，就形成了CMOS。NMOS晶体管和PMOS晶体管总是成对出现，状态互补。常用的CMOS门电路有反相器、与非门、或非门、与或非门、或与非门等。

1. CMOS反相器（CMOS非门）工作原理

图2-26显示了由NMOS和PMOS采用互补的方式组成的CMOS反相器的电路结构。图2-27是CMOS反相器当输入分别为低电平和高电平时的等效开关模型（注意，不是等效电路。提出等效开关模型是为了避开复杂的电路电气性能分析，直接了解其功能）。

图2-26中，设V_{IN}是输入电平，即作为此反相器的控制电压；V_{OUT}是反相器的输出电平；S_{TN}和S_{TP}分别是NMOS管和PMOS管。当图2-27（a）的输入V_{IN}为低电平L时（电压0V，对应逻辑0），上方的PMOS管导通（等效开关连通），下方的NMOS管截止（等效开关断开）。这时V_{OUT}的输出电压约等于5V，这时可定义为高电平H，即逻辑1。

反之，如图2-27（b）等效开关模型所示，反相器输入V_{IN}为高电平H时（如V_{DD}），上方的PMOS管截止（等效开关断开），下方的NMOS管导通（等效开关连通）。这时V_{OUT}的输出电压约0V，这时可定义此电压为低电平L，即逻辑0。

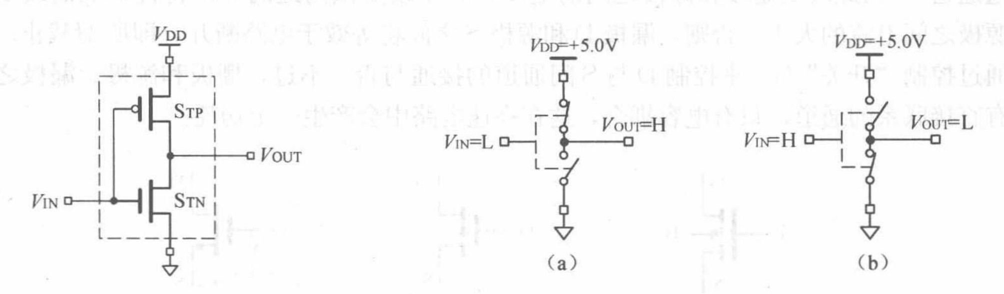
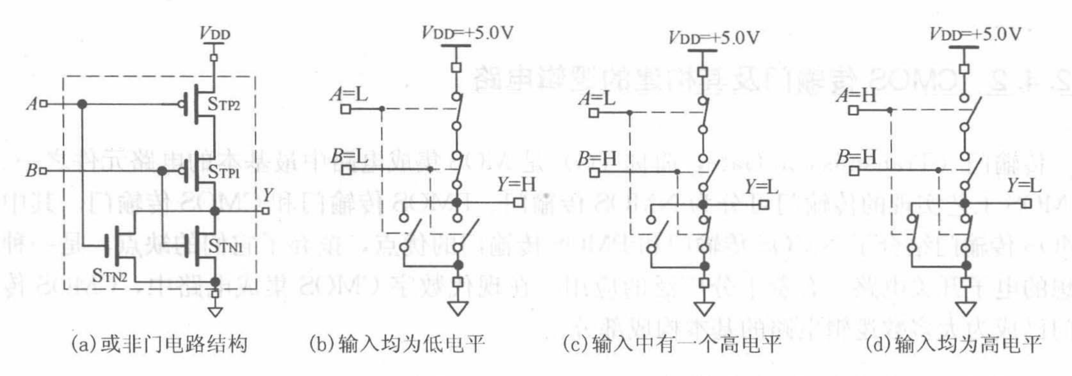


图2-26 CMOS反相器结构　　图2-27 CMOS反相器的等效开关模型

2. CMOS或非门工作原理

图2-28（a）是2输入CMOS或非门电路结构，电路由两对NMOS和PMOS采用互补方式组成。S_{TP1}和S_{TN1}组成一对互补结构，S_{TP2}和S_{TN2}组成另一对互补结构，其中S_{TN}是驱动管，S_{TP}是负载管。电路连接上采用驱动管并联连接、负载管串联连接的方式。图2-28（b）、（c）、（d）分别是在不同输入条件下的CMOS或非门的等效开关模型。表2-12是CMOS或非门逻辑真值表。



(a)或非门电路结构 (b)输入均为低电平 (c)输入中有一个高电平 (d)输入均为高电平

图 2-28 CMOS 或非门电路结构及其等效开关模型

从表 2-12 中可以看出，两个输入端 A、B 的输入电平可以有 4 种不同的组合，只有当 A、B 输入都是低电平时，输出 Y 才是高电平，而在其他情况下，只要输入端有一个或一个以上是高电平，输出端都输出低电平。因此，该电路实现的是或非门的功能。结合图 2-28（b）、（c）、（d）的等效开关模型，可以进一步证实表 2-12 的内容。

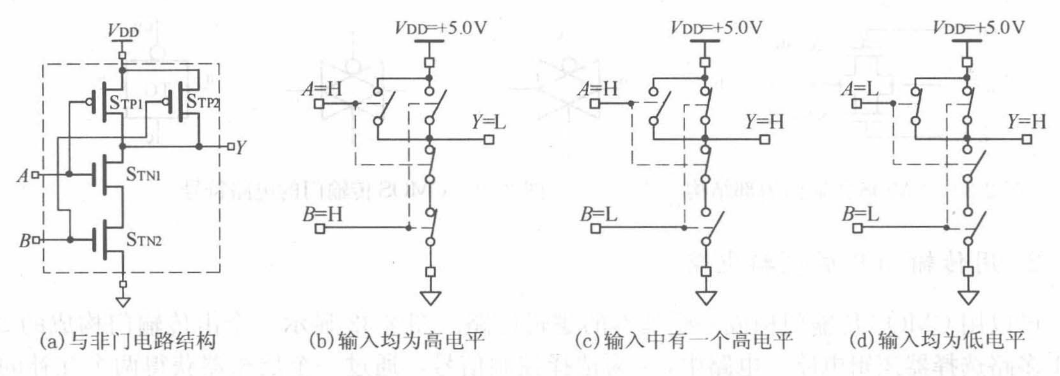
表 2-12 CMOS 或非门逻辑真值表

A	B	S_{TN1}	S_{TP1}	S_{TN2}	S_{TP2}	Y
L	L	off	on	off	on	H
L	H	off	on	on	off	L
H	L	on	off	off	on	L
H	H	on	off	on	off	L

3. CMOS 与非门工作原理

图 2-29（a）是 2 输入 CMOS 与非门电路结构，电路也由两对 NMOS 和 PMOS 采用互补方式组成。S_{TP1} 和 S_{TN1} 组成一对互补结构，S_{TP2} 和 S_{TN2} 组成另一对互补结构，其中 S_{TN} 是驱动管，S_{TP} 是负载管。电路连接上采用驱动管串联连接、负载管并联连接的方式。

图 2-29（b）、（c）、（d）分别是在不同输入条件下的 CMOS 与非门的等效开关模型。例如图 2-29（b）中，当输入端 A 和 B 都为高电平时，上方的两并联负载管的开关模型均呈断开状态，使输出端与电源隔绝，而下面的两串联驱动管的开关模型则呈闭合状态，即处于导通状态，从而使输出呈低电平；在其他情况下，输出都为高电平。显然该电路具有与非逻辑功能，即 $Y=\overline{AB}$。读者可以根据此电路模型写出其真值表。



(a)与非门电路结构 (b)输入均为高电平 (c)输入中有一个高电平 (d)输入均为低电平

图 2-29 CMOS 与非门电路结构及其开关模型

2.4.2 CMOS 传输门及其构建的逻辑电路

传输门（Transmission Gate，简称 TG）是 MOS 集成电路中最基本的电路元件之一。用 MOS 工艺实现的传输门可分为 NMOS 传输门、PMOS 传输门和 CMOS 传输门。其中 CMOS 传输门结合了 NMOS 传输门和 PMOS 传输门的优点，摈弃了它们的缺点，是一种理想的电子开关电路，有着十分广泛的应用。在现代数字 CMOS 集成电路中，CMOS 传输门已成为大多数逻辑电路的基本构成部分。

1. 传输门的结构和性能特点

CMOS 传输门是由一个 PMOS 和一个 NMOS 互补构成的，其内部结构简图如图 2-30 所示。图中的 PMOS 的源/漏极与 NMOS 的源/漏极直接连接，分别为 a 和 b 两个端口，PMOS 的栅极引出为 cn 端口，而 NMOS 的栅极引出为 c 端口。在正常使用时，要求满足 $cn=\bar{c}$，cn 是 c 的取反。为了便于理解，这里的 PMOS 和 NMOS 也可以直接看成是一个开关模型，当输入信号 c=1，cn=0 时，图中的 PMOS 和 NMOS 同时导通，a 端口和 b 端口就直接连接在了一起，即 a 上的电平可以直接传输到 b 端口，b 端口的电平也可以直接传输到 a 端口，构成了一个典型的双向通道。而当 c=0，cn=1 时，PMOS 和 NMOS 同时关闭，a 端口和 b 端口之间无法进行信号传输。

事实上，CMOS 传输门不仅仅是一个数字电路基本门，它还是一个模拟开关。从上面的描述可以发现，当 c=0 时，a 与 b 之间不导通，而 c=1 时，a 与 b 可双向导通，甚至可以用来传输模拟电压信号，该模拟电压值可以为 V_{DD} 与 V_{SS} 之间任意值。可以任意指定 a 为输入，或 b 为输入。如果 a 作为输入，当传输门导通时，b=a；当不导通时，则 b=高阻。

由于 PMOS 和 NMOS 在传输高电平和低电平时的特性是不同的，且正好相反，CMOS 传输门用 PMOS 和 NMOS 互补的结构很好地解决了这个问题，使得 CMOS 传输门在传输高电平或低电平的性能基本相同，这也是 CMOS 传输门使用两个开关 MOS 的原因。

CMOS 传输门的三种常用电路符号如图 2-31 所示。

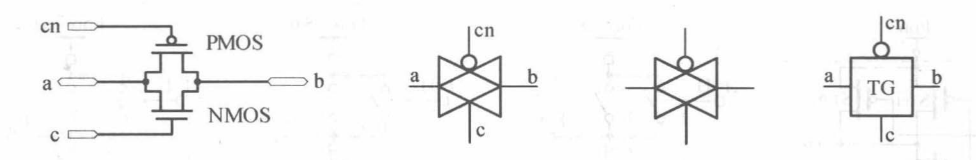


图 2-30 CMOS 传输门内部结构

图 2-31 CMOS 传输门的电路符号

2. 用传输门构成逻辑电路

可以用 CMOS 传输门构成一些基本的逻辑电路。图 2-32 显示一个由传输门构成的 2 选 1 多路选择器逻辑电路。电路中，s 为选择控制信号，通过一个反相器获得两个互补的控制信号，分别控制 TG1 和 TG2 两个传输门的控制端。当 s=0 时，TG1 处于导通状态，

而 TG2 处于截止状态，输出高阻状态，于是输出 y=a；反之，当 s=1 时，TG2 处于导通状态，而 TG1 处于输出高阻状态，于是有 y=b。这个逻辑关系可用逻辑表达式来描述，即 $y=\bar{s}\cdot a+s\cdot b$。

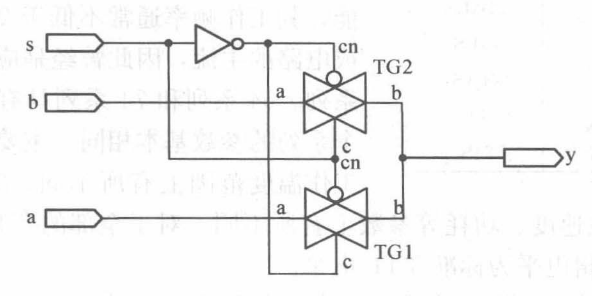


图 2-32 CMOS 传输门构成的 2 选 1 多路选择器

从电路结构上看，用 CMOS 传输门构建的 2 选 1 多路选择器，仅仅用了 6 个 MOS 管，要比使用与非门构成的同样功能的电路简洁许多，而且在实际电路中，电路中的反相器还可能被其他电路复用。其实，如果增加传输门的数量，修改传输门的控制逻辑，更复杂的 4 选 1 多路选择器也是很容易实现的。

2.4.3 TTL 集成电路逻辑门及同类 CMOS 器件系列

除了个别电路的端口驱动等情况外（如 74LS244、74LS245 等的应用），在当前的主流数字系统设计中，已很少使用 TTL 系列逻辑器件了。本节内容的主要目的只是希望借助曾经被广泛使用过的逻辑器件，使读者更好地理解集成逻辑电路的实用技术及数字技术的发展轨迹。

TTL 门电路由双极型三极管构成，其特点是，比 CMOS 型逻辑门电路的工作速度要快，抗静电能力强，但其功耗较大，不适宜做成大规模集成电路。此类器件曾广泛应用于小规模集成电路设计中，目前的应用范围已大为缩小。

TTL 门电路有 74（民用）和 54（军用）两大系列，每个系列中又有若干子系列。

例如，74 系列包含如下基本子系列：

- 74：标准 TTL（Standard TTL）。
- 74L：低功耗 TTL（Low-power TTL）。
- 74S：肖特基 TTL（Schottky TTL）。
- 74AS：先进肖特基 TTL（Advanced Schottky TTL）。
- 74LS：低功耗肖特基 TTL（Low-power Schottky TTL）。
- 74ALS：先进低功耗肖特基 TTL（Advanced Low-power Schottky TTL）。

每种集成电路又分为不同的系列，每个系列的数字集成电路都有不同的品种类型，用不同的代码表示，也就是器件型号的后几位数码。例如：74LS00 含有 4 个 2 输入与非门；7402 含有 4 个 2 输入或非门；74HC08 含有 4 个 2 输入与门；74LS27 含有 3 个 3 输入或非门；74LS86 含有 4 个 2 输入异或门，等等。

表 2-13 TTL 系列速度及功耗的比较

速度	TTL 系列	功耗	TTL 系列
最快	74AS	最小	74L
	74S		74ALS
	74ALS		74LS
	74LS		74AS
↓	74	↓	74
最慢	74L	最大	74S

数字电路设计者在选择 TTL 子系列时主要考虑它们的速度和功耗。其速度及功耗的比较见表 2-13。其中 74LS 系列产品具有较好的综合性能，其工作频率通常不低于 20MHz，是 TTL 集成电路的主流，因此曾经是应用最广的逻辑器件系列。54 系列和 74 系列具有相同的子系列，两个系列的参数基本相同，主要在电源电压范围和工作温度范围上有所不同。54 系列适应的范围更大些。不同子系列在速度、功耗等参数上有所不同。对于全部的 TTL 集成门器件都采用+5V 电源供电，逻辑电平为标准 TTL 电平。

与 TTL 门电路相比，CMOS 集成门电路（主要指 4000 和 4500 系列）的特点是集成度高、功耗低，但工作速度较慢（工作频率在 5MHz 左右）、抗静电能力差；其供电电源可以在 3～18V 之间很宽的范围内工作。不过它们的衍生产品，74HC 系列的 CMOS 门电路的工作速度有了很大的提高，抗静电能力也有了很大的改善，工作电压可与 TTL 门电路的逻辑电平兼容。另外，随着集成电路技术的不断提高，同 TTL 门电路一样，此类 CMOS 门电路也有 74 和 54 两大系列。此外，CMOS 门电路的工作电压也在不断下降，3.3V 或 2.5V 工作电压的 CMOS 门电路的应用也已十分广泛，它们的功耗比 5V CMOS 门电路低得多，所以在大规模集成电路和微处理器中已经占有重要地位。

74 系列 CMOS 门电路的基本子系列如下：

- 74HC 和 74HCT：高速 CMOS（High-speed CMOS），T 表示和 TTL 直接兼容。
- 74AC 和 74ACT：先进 CMOS（Advanced CMOS），它们提供了比 TTL 系列更高的速度和更低的功耗。
- 74AHC 和 74AHCT：先进高速 CMOS（Advanced High-speed CMOS）。

74 系列 3.3V CMOS 门电路的基本子系列有：

- 74LVC：低压 CMOS（Lower-voltage CMOS）。
- 74ALVC：先进低压 CMOS（Advanced Lower-voltage CMOS）。

此系列产品与 5V 电源电压工作下的 CMOS 集成电路（主要指 4000 和 4500 系列 CMOS 器件）相比，在工作速度相当的条件下，其功耗要减少约 34%。

TTL 类型的 74、54 系列器件和 CMOS 类型的 74HC，4000 和 4500 等器件统称为标准逻辑器件或通用逻辑器件，在传统的数字电路设计中十分常用。

2.4.4 集成电路门的性能参数

以下仅从实用的角度介绍集成逻辑门电路的几个外部特性参数，目的是希望对集成逻辑门电路的性能指标有一个概括性的认识。至于每种集成逻辑门的实际参数，可在具体使用时查阅有关的产品手册和说明，甚至直接通过实验测试获得。

数字集成电路的性能参数主要包括：直流电源电压、输入/输出逻辑电平、扇入/扇出系数、传输延时、功耗等。了解这些内容对顺利完成数字电路实验和设计十分重要。

1. 器件的工作电源电压

TTL集成电路的标准直流电源电压为5V，最低4.5V，最高5.5V。CMOS集成电路的直流电源电压可以在3～18V之间；74系列CMOS集成电路有5V和3.3V两种。CMOS电路的一个优点是电源电压的允许范围比TTL电路宽，如5V CMOS电路当其电源电压在2～6V范围内时都能正常工作。3.3V CMOS电路当其电源电压在2～3.6V范围内时仍能正常工作。

现代多种类型的逻辑器件或专用集成电路通常需要多个工作电源电压，它包括内核工作电源电压VCCINT、输入输出端口驱动电源电压VCCIO、特定核心模块工作电压（如嵌入式锁相环工作电压）等。VCCINT用于驱动逻辑器件的内部逻辑电路，而VCCIO则用于驱动此器件对外输入输出端口的电平控制电路设备。通常，VCCINT小于等于VCCIO，而同一器件的VCCIO也可以不同。这样有利于降低功耗、提高速度和兼容更多不同工作电压的接口器件。对于现代常用的逻辑器件，VCCIO小于等于3.3V；而VCCINT对于不同的器件有不同的要求，如有3.3V、2.5V、1.5V、1.2V或更低。一般来说，器件的集成度越高，它的内核电压VCCINT越低。

2. 逻辑器件的输入/输出逻辑电平

对于诸如TTL或CMOS集成门电路来说，它的输出"高电平"并不是理想的工作电源电压5V或3.3V电压；其输出"低电平"也并不是理想的0V电压。这主要是由于制造工艺上的离散性，使得即使是同一型号的器件，输出电平也不可能完全一样。另外，由于所带负载及环境温度等外部条件的不同，输出电平也会有较大的差异。但是，这种差异应该在一定的允许范围之内，否则就会无法正确标识出逻辑值1和逻辑值0，从而造成错误的逻辑操作或逻辑判断。

数字集成电路分别有如下4种不同的输入/输出逻辑电平。对于标准TTL电路有：

- 定义为逻辑0的低电平输入电压范围V_{IL}：0～0.8V。
- 定义为逻辑1的高电平输入电压范围V_{IH}：2～5V。
- 定义为逻辑0的低电平输出电压范围V_{OL}：不大于0.3V。
- 定义为逻辑1的高电平输出电压范围V_{OH}：不小于2.4V，

门电路输出高、低电平的具体电压值与所接的负载也有关系。

对于5V CMOS电路有：

- 定义为逻辑0的低电平输入电压范围V_{IL}：0～0.5V。
- 定义为逻辑1的高电平输入电压范围V_{IH}：2.5～5V。
- 定义为逻辑0的低电平输出电压范围V_{OL}：不大于0.1V。
- 定义为逻辑1的高电平输出电压范围V_{OH}：不小于4.4V。

以上这些数据并非是绝对的，对于具体的器件，还应该根据实际情况来确定，特别是目前陆续出现的一些新的逻辑器件。

如图2-33所示，当输入电平在$V_{IL(max)}$和$V_{IH(min)}$之间时，逻辑电路可能把它当作0，也可能把它当作1，这是一个逻辑电平不稳定或不确定的电平区域。此外，当逻辑电路因所

接负载过多等原因不能正常工作时，高电平输出可能低于 $V_{OH(min)}$，低电平输出可能高于 $V_{OL(max)}$ 等现象，也都不是正常的现象，这都应该注意避免。

注意： 图 2-33 给出的电平定义只具有相对意义，不同类型的数字器件有不同的电平定义，这需要参考相关的技术资料。

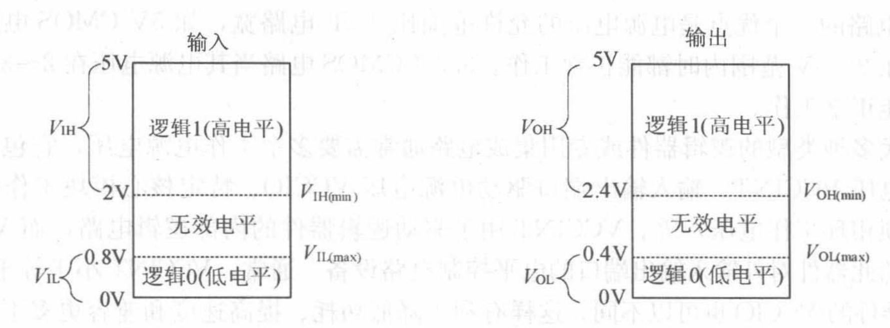


图 2-33　标准 TTL 门的输入/输出逻辑电平

3. 逻辑信号传输延迟时间

在集成门电路中，由于晶体管开关时间的影响，使得输出与输入之间存在信号的传输延迟。显然，传输延时越短，工作速度就越快，工作频率也越高。因此，传输延迟时间是衡量门电路工作速度的重要指标。例如，传输时间为 10ns 的逻辑电路要比 20ns 的电路快一倍。

由于实际的信号波形有上升沿和下降沿之分，因此传输延迟时间 t_{pd} 是两种变化情况所反映的结果。一是输出从高电平转换到低电平时，输入脉冲指定参考点与输出脉冲的对应参考点之间的时间，记为 t_{PHL}；另一种是输出从低电平转换到高电平时的情况，记作 t_{PLH}。如图 2-34 所示是一个反相器的传输延迟时间 t_{PHL} 和 t_{PLH} 的测量。参考点可以选在输入和输出脉冲相应边沿的 50％处。在实际测试中常用平均传输延迟时间来表示门电路的传输延迟这一指标：

$$t_{pd}=\frac{1}{2}(t_{PHL}+t_{PLH})$$

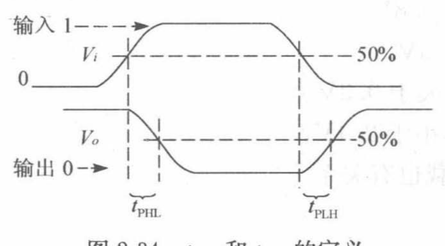


图 2-34　t_{PHL} 和 t_{PLH} 的定义

TTL 集成门电路的传输延迟时间 t_{pd} 的值为几十纳秒至几百纳秒。传统 CMOS 集成门电路的传输延迟时间 t_{pd} 较大，有上百纳秒。但高速 CMOS 系列的 t_{pd} 较小，仅几纳秒左右；ECL 集成门电路的传输延迟时间 t_{pd} 更小些。

4. 集成逻辑电路的扇入和扇出系数

对于集成逻辑门电路，驱动门与负载门之间的电压和电流关系如图 2-35 所示，这实际上是电流在一个逻辑电路的输出与另一个电路的输入之间如何流动的描述。

在高电平输出状态下，驱动门提供电流 I_{OH} 给负载门，作为负载门的输入电流 I_{IH}。这时驱动门处于“拉电流”工作状态。拉电流越大，输出端的高电平就越低。这是因为输出

级，即驱动门内部存在内阻，而大电流将在内阻上有更大的压降，从而造成输出电压下降。然而从图 2-33 可知，输出的高电平是有一定限制的，它有一个最小值 $V_{OH(min)}$。对于 TTL 器件，这个值应该大于 2.4V。显然，驱动门的输出电流是有一定限度的，即其 I_{OH} 对应于 $V_{OH(min)}$，驱动门应该有一个所能承受的最大值，即 $I_{OH(max)}$。当负载门是同类型逻辑门时，接在驱动门输出端的负载门的输入端越多，则 I_{OH} 将越大。

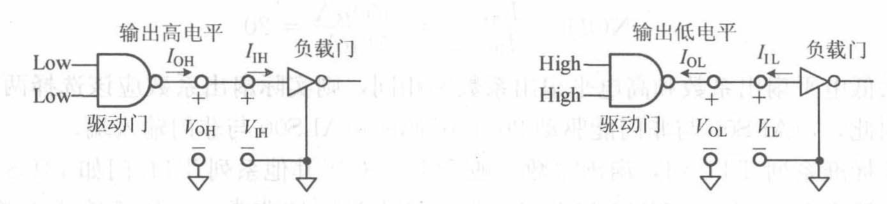


图 2-35 两种逻辑状态中的电流和电压

另一方面，在驱动门的低电平输出状态下，来自负载门输入端的电流 I_{IL} 将倒灌进驱动门的输出端，使之处于“灌电流”状态。同样，驱动门的输出端能够承受的灌电流的大小也是有限的。因为随着灌电流 I_{OL} 的加大，驱动门的内阻会使输出端的电平相应地升高。然而图 2-33 显示，输出的低电平有一个最大值，即 $V_{OL(max)}$，它将使驱动门的灌电流对应一个所能承受的最大值，即 $I_{OL(max)}$。

显然，$I_{OH(max)}$ 和 $I_{OL(max)}$ 的大小标志着逻辑门器件的驱动能力。为此，需要定义某个指标来描述逻辑门的驱动能力。扇入和扇出系数反映了门电路的输入端数目和输出驱动能力的指标。

扇入系数：指一个门电路所能允许的输入端个数。

扇出系数：指一个门电路所能驱动的同类门电路输入端的最大数目。

扇出系数越大，门电路的带负载能力就越强。一般来说，CMOS 电路的扇出系数比 TTL 电路高。

定义逻辑门输出高电平时的扇出系数 NOH 的计算公式为

$$\mathrm{NOH} = \frac{I_{OH(max)}}{I_{IH(max)}}$$

其中的 $I_{IH(max)}$ 是进入单个同类门一个输入端的最大漏电流。

定义逻辑门输出低电平时的扇出系数 NOL 的计算公式为

$$\mathrm{NOL} = \frac{I_{OL(max)}}{I_{IL(max)}}$$

其中的 $I_{IL(max)}$ 是从单个同类门一个输入端（流出）的短路电流，也即将此端接地的电流。

通常，NOH 和 NOL 的大小是不一致的。对于不同的器件，NOH 和 NOL 间的差异也不同，因此，描述驱动门或逻辑门的实际扇出系数，应该取 NOH 和 NOL 中的最小值。

【例 2-1】 已知 74ALS00 的电流参数为 $I_{OL(max)}=8\text{mA}$，$I_{IL(max)}=0.1\text{mA}$，$I_{OH(max)}=0.4\text{mA}$，$I_{IH(max)}=20\mu\text{A}$。求一个 74ALS00 与非门输出能驱动多少个 74ALS00 与非门的输入。

解：首先考虑低电平状态。在低电平状态下得到能被驱动的输入个数：

$$NOL = \frac{I_{OL(max)}}{I_{IL(max)}} = \frac{8mA}{0.1mA} = 80$$

注意： 在查IC手册时，我们会发现输入电流 I_{IL} 实际上是 $-0.1mA$。这里的负号用来表示电流是由输入端流出的。今后，在计算中可以忽略负号。

在高电平状态能驱动的输入个数是

$$NOH = \frac{I_{OH(max)}}{I_{IH(max)}} = \frac{400\mu A}{20\mu A} = 20$$

如果低电平扇出系数和高电平扇出系数不相同，则实际扇出系数应该选择两个中的较小者。因此，74ALS00与非门能驱动20个其他的74ALS00与非门输入端。

对于标准系列TTL门，扇出系数一般为10；对于其他系列TTL门如74LS系列，扇出系数一般为20。对于CMOS门电路，虽然输入端阻抗非常高，所需输入电流非常小，但由于其输入端有分布电容，当电平发生变化时，电容有充放电电流通过。因此，CMOS门电路输出端可接的输入端数量也是受到限制的，其扇出系数一般为50左右。

在以上已经做了讨论，当输入端个数超过扇出系数时，就可能改变原来的输出电平，使得输出的低电平超过 $V_{OL(max)}$，或者输出高电平低于 $V_{OH(min)}$，从而导致输出电平产生混乱，乃至输出的逻辑信息产生混乱。为了解决这些问题，可接入缓冲门增大输出端的驱动能力，以避免混乱电平（无效电平）的出现。

注意： 在过去传统的小规模逻辑电路设计中，扇入扇出问题常成为一个必须注意的问题。但在现代的数字系统设计中最多只在约束设置时考虑。这是因为现在设计的逻辑电路的规模很大，且趋向于单片实现方案，因而可以很好地使用自动化设计技术，使扇入扇出问题在设计软件中被自动考虑进去，不必人为介入了。

5. 集成逻辑门器件的功耗

功耗是指门电路通电工作时所消耗的电功率，它等于电源电压 V_{CC} 和电源电流 I_{CC} 的乘积，即功耗 $P_D = V_{CC} \cdot I_{CC}$。但由于在门电路中电源电压是固定的，而电源电流不是常数，也就是说，在门电路输出高电平和输出低电平时通过电源的电流是不一样的，因而这两种情况下的功耗大小也不一样。一般求它们的平均值（I_{CCH} 和 I_{CCL} 分别是门电路输出高电平和输出低电平时通过的电源电流）：

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

一般情况下，CMOS集成电路的功耗较低，而且与工作频率有关，频率越高功耗越大，其数量级为微瓦，因而CMOS集成电路广泛应用于电池供电的便携式产品中；TTL集成电路的功耗较高，其数量级为毫瓦，且基本与工作频率无关。

2.4.5 TTL与CMOS集成电路的传统接口技术

由于TTL门电路和CMOS门电路是两种不同类型的电路，它们的参数并不完全相同。因此，在传统的数字电路设计中，如果同时使用TTL门电路和CMOS门电路，为了

保证系统能够正常工作，必须考虑两者之间的连接问题，以满足表 2-14 所列条件。

如果不满足表 2-14 所列条件，必须增加接口电路。常用的方法有增加上拉电阻、采用专门接口电路、驱动门并接等。如图 2-36 所示，这是 TTL 门驱动 CMOS 门的情况，为了两者的电平匹配，在 TTL 驱动门的输出端接了上拉电阻 R。

表 2-14 TTL 门与 CMOS 门的连接条件

驱动门		负载门
$V_{OH(min)}$	>	$V_{IH(min)}$
$V_{OL(max)}$	<	$V_{IL(max)}$
I_{OH}	>	I_{IH}
I_{OL}	>	I_{IL}

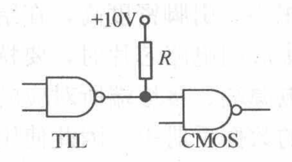


图 2-36 TTL 驱动门与 CMOS 负载门的连接

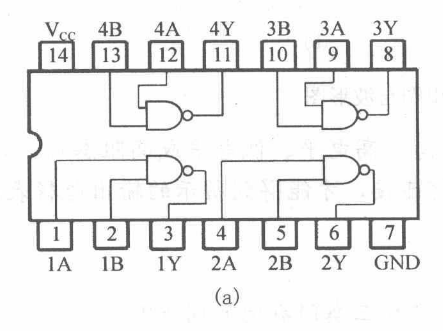
凡是和 TTL 门兼容的 CMOS 门（如 74HCTXX 和 74ACTXX 系列 CMOS 门）可以和 TTL 的输出端直接连接，不必外加元器件。至于其他 CMOS 门电路与 TTL 门电路的连接，可以采用电平转换器，如 CC4049（六反相器）或 CC4050（六缓冲器）等，或采用 CMOS 漏极开路门（OD 门），如 CC40107 等，其具体方法可以参考相关的技术资料。

由于 TTL 与 CMOS 门混合应用易导致系统速度和可靠性的降低（如上拉电阻导致的分布电容增加等），且现代数字系统主要由大规模 PLD 或用单片大规模 IC 器件实现，故在自动化数字系统设计中已不存在 TTL 与 CMOS 门混合应用的问题了。

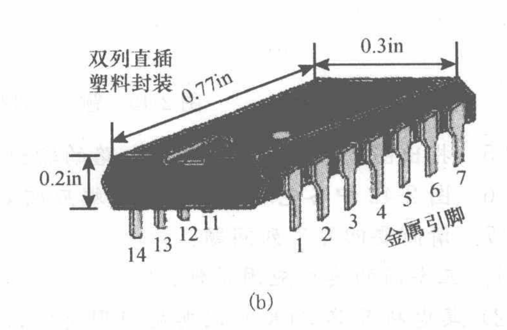
2.4.6 CMOS 与 TTL 逻辑器件的封装

相同品种类型编号的标准逻辑集成电路，不论属于哪个系列，它们的逻辑功能都相同，外形尺寸也相同（即封装相同），引脚也兼容；只是工作速度、功耗、工作电源电压范围等参数有所不同。例如 7400、74LS00、74ALS00、74HC00、74AHC00 都是 14 个引脚兼容的 4 路 2 输入与非门封装。

图 2-37 给出了 74LS00 芯片的引脚图和封装图。从图 2-37（a）可以看出，此器件内含有 4 个与非门。第 14 脚是工作电源输入端，接 5V 电源电压；第 7 脚接地，接电源地，电平为 0V。图 2-37（b）是此器件的外形图，即封装图。其封装属于双列直插式封装 DIP(Dual In-line Package)，是采用塑料或陶瓷封装技术的双列直插式封装，这种封装绝



(a)



(b)

图 2-37 74LS00 引脚配置及 DIP 封装外形图

缘密封，其强度和耐高温性能都比较好，且利于直接插到电路板上。这曾经是最常用的封装形式。但由于其体积大，可向外引的引脚数太少等缺点，此类封装的器件已经很少在现代电子系统设计中应用了。其他型号芯片的封装和引脚图可参考相关数据手册。

常见的另一种 IC 封装形式是 SMT（Surface-Mount Technology）封装，简称表面贴装。SMT 封装的芯片直接焊接在电路板的表面，而无需在印刷电路上穿孔，所以其集成度高，面积小，引脚密度高，在给定区域内可以放置更多的集成电路芯片。

使用集成门电路芯片时，要特别注意其引脚配置及排列情况，分清每个门的输入端、输出端和电源端、接地端所对应的引脚，这些信息及芯片中门电路的性能参数，都收录在有关产品的数据手册中，因此使用时要养成常查数据手册的习惯。

习　题

2-1　试分别画出实现下列逻辑功能的 CMOS 电路图。

（1）$F=\overline{A+B+C}$　（2）$F=\overline{A\cdot B\cdot C}$　（3）$F=\overline{A+BC}$

2-2　试画出图 2-38（a）中门电路 F_1、F_2、F_3 的输出波形，输入 A、B 的波形如图 2-38（b）所示。

2-3　求图 2-39 所示电路的输出逻辑表达式。

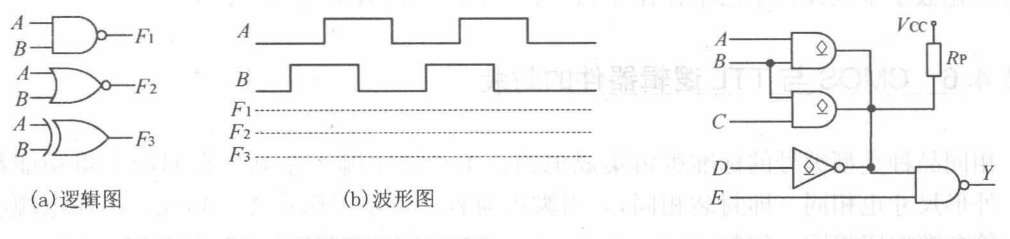


图 2-38　题 2-2 的逻辑图与波形图　　图 2-39　题 2-3 逻辑图

2-4　门电路符号和 A、B、C 输入波形如图 2-40 所示，根据给定的输入波形画出输出波形。

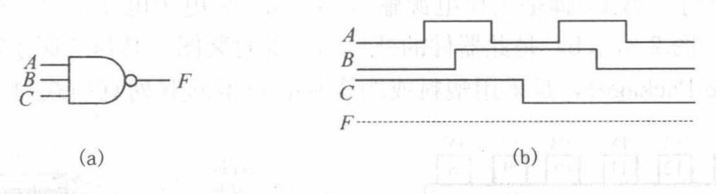


图 2-40　题 2-3 的逻辑图与波形图

2-5　指出图 2-41 中各 TTL 门电路的输出状态（高电平、低电平或高阻态）。

2-6　图 2-42 中各电路的每个输入端应该如何连接，才能得到所示的输出逻辑表达式？

2-7　请简要回答下列问题：

（1）三态门的典型应用是什么？

（2）集电极开路（OC）门典型应用是什么？它和三态门有何异同点？

（3）或非门无用输入端应如何处理？

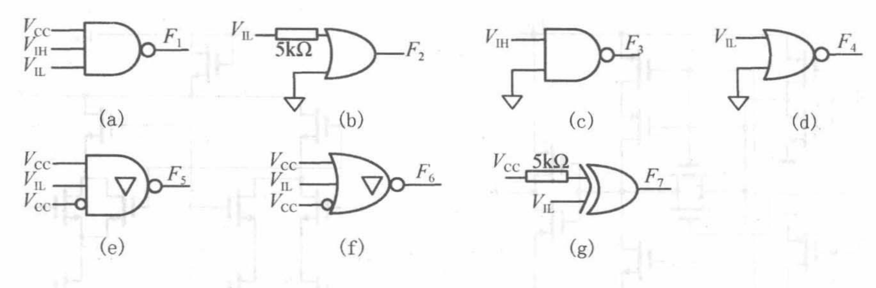


图 2-41 题 2-5 的逻辑图

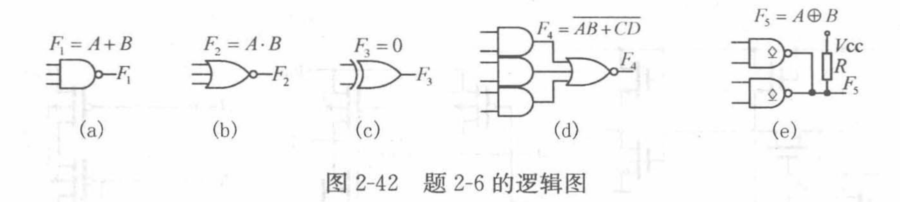


图 2-42 题 2-6 的逻辑图

2-8 试分别写出图 2-43 中二 NMOS 门电路 F_1、F_2 的输出逻辑表达式。

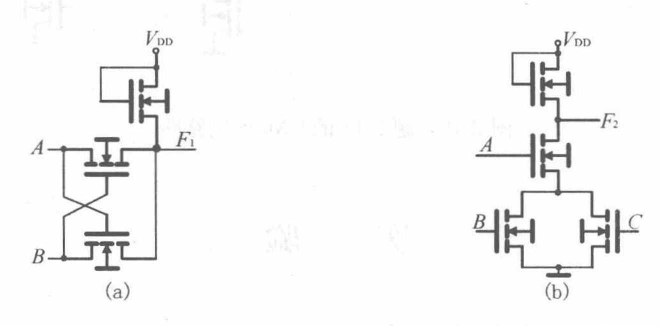


图 2-43 题 2-8 电路图

2-9 集成逻辑电路 74LS245 的内部部分结构如图 2-44 所示，试说明该电路的逻辑功能。

2-10 集成逻辑电路 74LS244 的内部部分结构如图 2-45 所示，试说明该电路的逻辑功能。

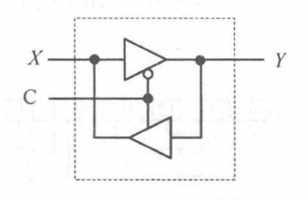


图 2-44 题 2-9 的逻辑结构

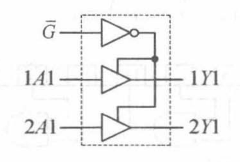


图 2-45 题 2-10 的逻辑结构

2-11 试分别写出如图 2-46 所示的各 CMOS 器件的电路图中输出信号 L 或 Y 的逻辑表达式，并说明该逻辑电路的功能。

2-12 试用 CMOS 传输门分别绘出与非门、或非门和异或门的逻辑电路。

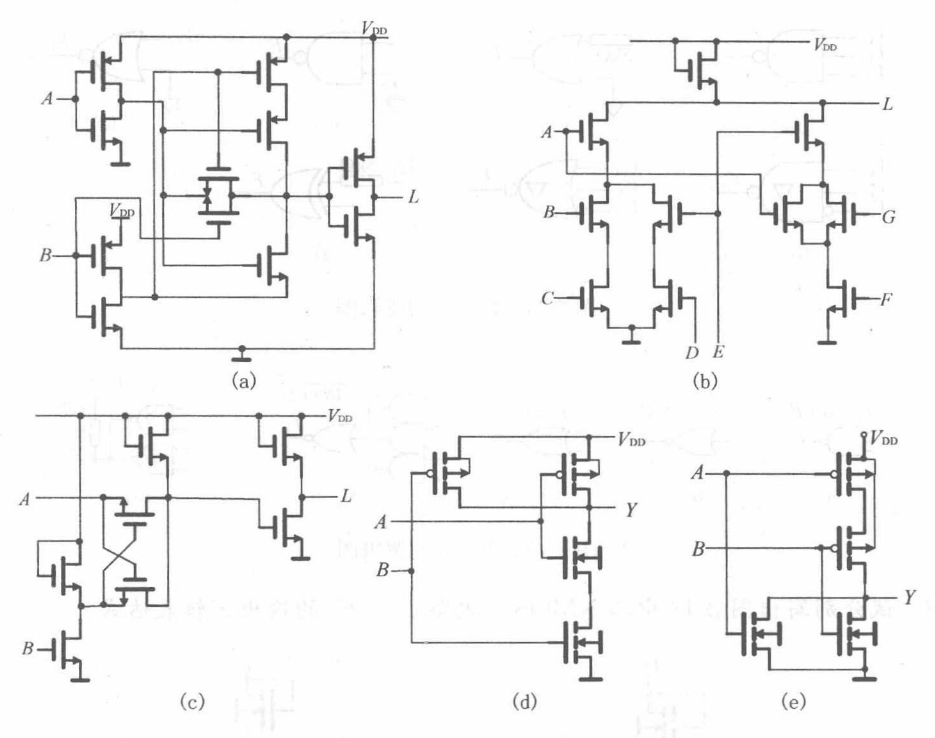


图 2-46　题 2-11 的 CMOS 电路图

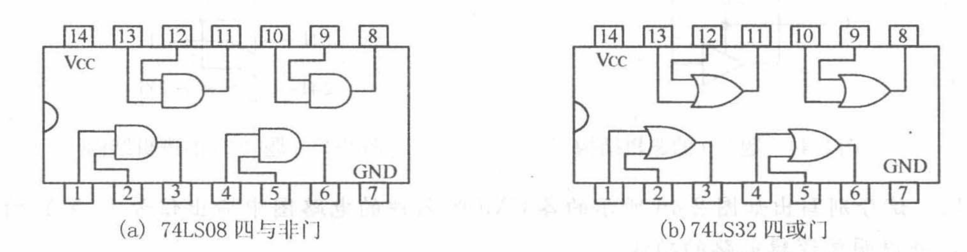


2-1　集成电路 TTL 和 CMOS 器件的逻辑功能和性能参数测试

根据 2.4 节的原理，分别测试图 2-47 的 TTL 器件和 CMOS 器件的逻辑功能和性能参数。测试内容包括：

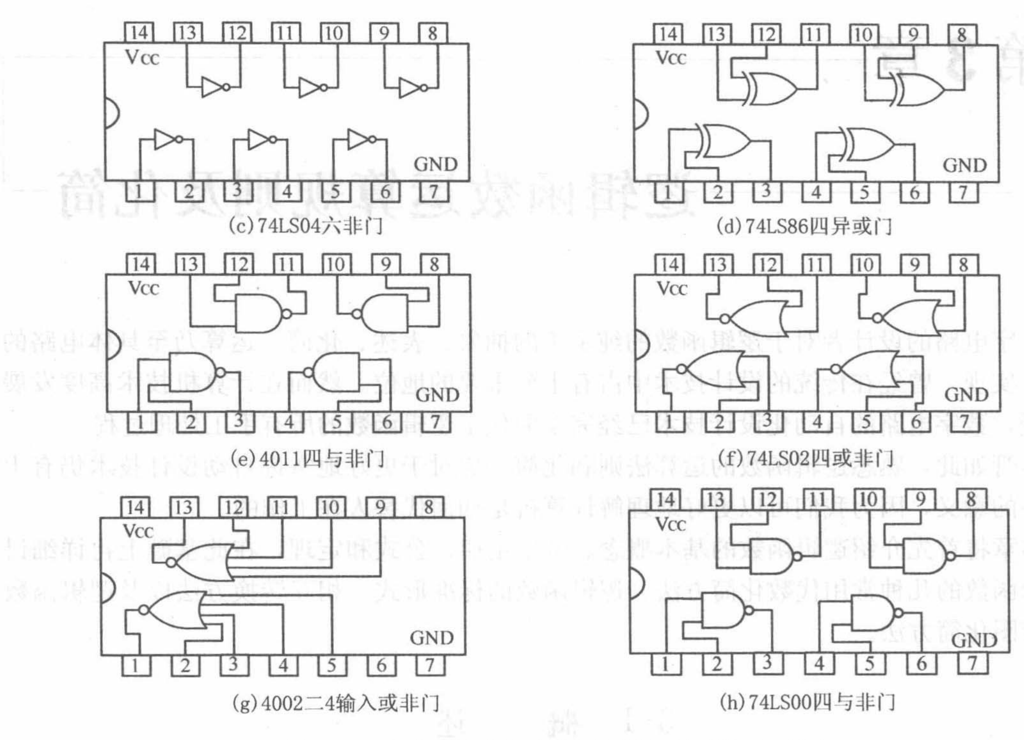
(1) 逻辑功能测试：在输入端输入高、低电平信号的不同组合，测出相应的输出逻辑电平。

(2) 集成电路门的性能参数测试：分别测试标准 TTL 门和 CMOS 门的输入/输出逻辑电平 V_{IL}、V_{IH}、V_{OL}、V_{OH}。



(a) 74LS08 四与非门　(b) 74LS32 四或门

图 2-47　集成逻辑器件内部逻辑结构及引脚图



(c) 74LS04六非门 (d) 74LS86四异或门

(e) 4011四与非门 (f) 74LS02四或非门

(g) 4002二4输入或非门 (h) 74LS00四与非门

图 2-47 集成逻辑器件内部逻辑结构及引脚图（续）

（3）比较标准 TTL 器件和 CMOS 器件的性能特点，总结与门、或门、非门、与非门、或非门、异或门的逻辑规律。完成实验报告。

第 3 章 逻辑函数运算规则及化简

数字电路的设计者对于逻辑函数的纯手工的抽象、表述、化简、运算乃至具体电路的实现，曾经在传统的设计技术中占有十分重要的地位。然而在计算机技术高度发展的今天，数字电路的自动化设计技术已经完全取代了逻辑函数的所有手工处理过程。

尽管如此，熟悉逻辑函数的运算法则和化简方法对于更好地掌握自动设计技术仍有十分重要的意义，因为我们可以更好地理解计算机是如何代替人在工作的。

本章将首先介绍逻辑函数的基本概念、基本定律、公式和定理，在此基础上再详细讨论逻辑函数的几种常用代数化简方法、逻辑函数的标准形式、相互转换方法以及逻辑函数的卡诺图化简方法。

3.1 概　　述

逻辑函数（或逻辑代数），也叫布尔函数，是英国数学家乔治·布尔（George Boole）于 1849 年首先提出的，它是进行逻辑运算的最基本的数学方法。逻辑代数已成为分析和设计数字电路不可缺少的数学工具。

正如第 2 章所介绍的，数字电路是一种开关电路，其开关的两种状态——“开通”与“关断”，可用二元或二值常量 0 和 1 来表示。对应于具体的数字电路，其输入、输出量值，一般用高、低电平来体现。高低电平又可抽象为二元常量。如用“1”表示高电平，用“0”表示低电平。显然，二值常量 1 和 0 不仅可以用来表示二进制数值，还可以表示许多对立的逻辑状态。由于数字电路的输入输出量之间存在确定的逻辑对应关系，所以完全可以用逻辑函数来描述数字电路输入输出间的这种因果关系。

逻辑代数则是提供了一种方法，即使用二值函数进行逻辑运算，使得本来用语言描述显得十分复杂的逻辑命题，在使用逻辑代数语言后，就变成了简单的代数表达式，因此被称为“逻辑函数”。一般意义的逻辑函数的表示方法如下：

设输入逻辑变量为 A、B、C、…，输出逻辑变量为 F。当 A、B、C、…的取值确定后，F 的值就被唯一地确定下来，则称 F 是 A、B、C、…的逻辑函数，记为

$$F = f(A,B,C,\cdots)$$

注意： 传统的纯抽象的逻辑变量和逻辑函数的取值只能是 0 或 1，没有其他中间值。但自从出现了 HDL，逻辑变量取值的范围有了扩充，除了 0 和 1 外还有高阻态等。

逻辑函数可以分别用真值表、逻辑表达式、逻辑图、波形图和卡诺图来表示。各种表示方法具有不同的特点和用途。

3.2 逻辑代数的运算规则

以下首先介绍逻辑代数运算的交换律、结合律和分配律，逻辑代数的基本公式，摩根定理及其不同形式，然后给出逻辑代数的代入规则、反演规则和对偶规则，以及这些公式的使用方法。

3.2.1 逻辑代数的基本公理

逻辑代数有以下5个基本公理：

公理1：设A为逻辑变量，若$A\neq 0$，则$A=1$；若$A\neq 1$，则$A=0$。

这个公理决定了逻辑变量的双值性。在逻辑变量和逻辑函数中的0和1，不是数值的0和1，而是代表两种逻辑状态。

公理2：$0\cdot 0=0$；$1+1=1$。

式中的点表示逻辑与，在用文字表述时可以省略；加号表示逻辑或。

公理3：$1\cdot 1=1$；$0+0=0$。

公理4：$0\cdot 1=0$；$1+0=1$。$1\cdot 0=0$；$0+1=1$

公理5：$\overline{0}=1$；$\overline{1}=0$。

3.2.2 逻辑代数的基本定律

和普通代数一样，逻辑代数有如下基本定律：

(1) **0-1律**：$A\cdot 0=0$；$A+1=1$。

(2) **自等律**：$A\cdot 1=A$；$A+0=A$。

(3) **重叠律**：$A\cdot A=A$；$A+A=A$

(4) **互补律**：$A\cdot\overline{A}=0$；$A+\overline{A}=1$。

(5) **还原律**：$\overline{\overline{A}}=A$。

(6) **交换律**：$A\cdot B=B\cdot A$；$A+B=B+A$。

(7) **结合律**：$A\cdot(B\cdot C)=(A\cdot B)\cdot C$；$A+(B+C)=(A+B)+C$。

以上各定律均可用公理来证明，方法是将逻辑变量分别用0和1代入，所得的表达式符合公理2～公理5。

(8) **分配律**：$A\cdot(B+C)=AB+AC$；$A+(B\cdot C)=(A+B)\cdot(A+C)$。

加（逻辑或）对乘（逻辑与）的分配律证明如下：

$$\begin{aligned}A+(B\cdot C)&=A(1+B+C)+BC &&\text{（利用0-1律和自等律）}\\&=A+AB+AC+BC &&\text{（利用乘对加的分配律）}\\&=AA+AB+AC+BC &&\text{（利用重叠律）}\\&=A(A+B)+C(A+B) &&\text{（利用乘对加的分配律）}\\&=(A+B)(A+C) &&\text{（利用乘对加的分配律）}\end{aligned}$$

(9) **吸收律**：$A+A\cdot B=A$；$A\cdot(A+B)=A$。

证明：$A+A\cdot B=A(1+B)=A\cdot 1=A$

$A\cdot(A+B)=AA+AB=A+AB=A(1+B)=A$

(10) **等同律**：$A+\overline{A}B=A+B$；$A\cdot(\overline{A}+B)=AB$。

证明：$A+\overline{A}B=A(1+B)+\overline{A}B=A+AB+\overline{A}B=A+B(A+\overline{A})=A+B$

(11) **反演律**（摩根定理）：$\overline{A\cdot B}=\overline{A}+\overline{B}$；$\overline{A+B}=\overline{A}\cdot\overline{B}$。

也可用其他方法来证明这些逻辑定律，如采用真值表来证明。

以下真值表（表3-1）的证明可见反演律成立。

表3-1 真值表

A	B	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$	$\overline{A+B}$	$\overline{A}\cdot\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	0	0	0

(12) **包含律**：$AB+\overline{A}C+BCD=AB+\overline{A}C$。

证明：
$$\begin{aligned}AB+\overline{A}C+BCD&=AB+\overline{A}C+BCD(A+\overline{A})\\&=AB+\overline{A}C+ABCD+\overline{A}BCD\\&=(AB+ABCD)+(\overline{A}C+\overline{A}BCD)\\&=AB(1+CD)+\overline{A}C(1+BD)=AB+\overline{A}C\end{aligned}$$

包含律表明，在含有互反因子的两个积项中，如果除去这两个互反因子外，其他因子均为另一个积项的因子，则另一个积项就是多余的，便可以消掉。

结合本例可以看到：在含有互反因子 A 和 $\overline{A}$ 的两个积项 AB 和 $\overline{A}C$ 中，除了 A 和 $\overline{A}$ 两个因子外，其余因子 B 和 C 均是另一个积项 BCD 的因子。则 BCD 积项是多余的，便可消掉，不影响原逻辑关系。

3.2.3 摩根定理

摩根是与布尔同一时代的英国数学家，他提出了两条逻辑定理。它们成为逻辑表达式变换的强有力工具，是逻辑代数的重要组成部分，其主要内容如下：

(1) 逻辑变量与运算后取反等于各个逻辑变量分别取反的或运算。

用公式表示如下：

$$\overline{AB}=\overline{A}+\overline{B}$$

(2) 逻辑变量或运算后取反等于各个逻辑变量分别取反的与运算。

用公式表示如下：

$$\overline{A+B}=\overline{A}\,\overline{B}$$

上述两个定理也适用于多个变量的情形，如：

$$\overline{ABC}=\overline{A}+\overline{B}+\overline{C}\qquad\overline{A+B+C}=\overline{A}\,\overline{B}\,\overline{C}$$

【**例3-1**】应用摩根定理化简逻辑函数 $F=\overline{(AB+\overline{C})(A+\overline{B}C)}$。

解：反复应用摩根定理可得

$$F=\overline{AB+\overline{C}}+\overline{A+\overline{B}C}=\overline{AB}C+\overline{A}\,\overline{\overline{B}C}$$
$$=(\overline{A}+\overline{B})C+\overline{A}(B+\overline{C})=\overline{A}C+\overline{B}C+\overline{A}B+\overline{A}\,\overline{C}=\overline{A}+\overline{B}C$$

3.2.4 逻辑代数的基本规则

以下给出的 3 个运算规则可以更好地扩展以上的定理在逻辑代数运算中的范围。

1. 代入规则

任何一个含有变量 A 的逻辑等式，如果将所有出现 A 的位置都代之以同一个逻辑函数 F，则等式仍然成立。这个规则称为代入规则。

例如，给定逻辑等式 $A(B+C)=AB+AC$，若等式中的 C 都用（$C+D$）代替，则该逻辑等式仍然成立，即：$A(B+(C+D))=AB+A(C+D)$。

代入规则可以用来扩展定理的应用范围，因为将已知等式某一个变量用任意一个函数代替后，就得到一个新的等式。

2. 反演规则

对于任何一个逻辑表达式 F，若将其中所有的与“·”换成或“+”，“+”换成“·”，“0”换成“1”，“1”换成“0”，原变量换成反变量，反变量换成原变量，则得到的结果就是 $\overline{F}$ 。这个规则叫做反演规则。

反演规则为求取已知逻辑函数的反函数提供了方便。但是在使用反演规则时应注意遵守以下两个原则：

（1）注意保持原函数中的运算符号的优先顺序不变。

【例 3-2】 已知逻辑函数 $F=\overline{A}+\overline{B}(C+\overline{D}E)$，试求其反函数。

解：$\overline{F}=A(B+\overline{C}(D+\overline{E}))$，而不应该是 $\overline{F}=AB+\overline{C}D+\overline{E}$。

（2）不属于单个变量上的反号应保留不变。或不属于单个变量上的反号下面的函数当一个变量处理。

【例 3-3】 已知 $F=A+\overline{B+\overline{C}\cdot\overline{D+\overline{E}}}$，求 $\overline{F}$。

解法一：$F=A+\overline{B+\overline{C}\cdot\overline{D+\overline{E}}}=A+\overline{B+\overline{C}\,\overline{D}E}$

$\overline{F}=\overline{A}(B+\overline{C}\,\overline{D}E)=\overline{A}B+\overline{A}\,\overline{C}\,\overline{D}E$

解法二：$\overline{F}=\overline{A+\overline{B+\overline{C}\cdot\overline{D+\overline{E}}}}=\overline{A}\cdot\overline{\overline{B}(C+D+\overline{E})}=\overline{A}(B+\overline{C+D+\overline{E}})$

$=\overline{A}(B+\overline{C}\,\overline{D}E)=\overline{A}B+\overline{A}\,\overline{C}\,\overline{D}E$

3. 对偶规则

对于任何一个逻辑表达式 F，如果将式中所有的“·”换成“+”，“+”换成“·”，“0”换成“1”，“1”换成“0”，而变量保持不变，原表达式中的运算优先顺序不变。那么

就可以得到一个新的表达式，这个新的表达式称为 F 的对偶式 F^*。

这个规则叫做对偶规则。

【例 3-4】 已知 $F=AB+\overline{C}D$，求 F^*。

解：$F^*=(A+B)(\overline{C}+D)$

【例 3-5】 已知 $F=A+\overline{B+\overline{C}\cdot\overline{D+\overline{E}}}$，求 F^*。

解：$F=A+\overline{B+\overline{C}\cdot\overline{D+\overline{E}}}=A+\overline{B+\overline{C}\,\overline{D}E}$

$F^*=A\cdot\overline{B(\overline{C}+\overline{D}+E)}=A\overline{B}+ACD\overline{E}$

对偶式有两个重要的性质。

性质 1：若 $F(A，B，C，\cdots)=G(A，B，C，\cdots)$，则 $F^*=G^*$。

性质 2：$(F^*)^*=F$。

根据对偶性质，当已证明某两个逻辑表达式相等时，便可知它们的对偶式也相等。显然利用对偶的性质可以使前面的定律、公式的数目减少一半。

有些逻辑函数表达式的对偶式就是原函数本身，即 $F^*=F$。这时称函数 F 为自对偶函数。

【例 3-6】 证明函数 $F=(A+\overline{C})\overline{B}+A(\overline{B}+\overline{C})$ 是一自对偶函数。

证明：

$$
\begin{aligned}
F^* &=(A\overline{C}+\overline{B})(A+\overline{B}\,\overline{C})=(A+\overline{B})(\overline{C}+\overline{B})(A+\overline{B})(A+\overline{C})\\
&=(A+\overline{B})(\overline{B}+\overline{C})(A+\overline{C})=A(\overline{B}+\overline{C})(A+\overline{C})+\overline{B}(\overline{B}+\overline{C})(A+\overline{C})\\
&=(\overline{B}+\overline{C})(A+A\overline{C})+(\overline{B}+\overline{B}\,\overline{C})(A+\overline{C})\\
&=A(\overline{B}+\overline{C})+\overline{B}(A+\overline{C})=F
\end{aligned}
$$

3.3 逻辑函数表述方法

当输入逻辑变量的值确定以后，输出变量的值也随之唯一地确定下来。因此输入变量和输出变量之间存在一种对应关系，这种逻辑对应关系称为逻辑函数。逻辑函数有各种不同的表述形式。如前所述，逻辑函数的表述方法有逻辑代数表达式、逻辑图、逻辑真值表、卡诺图、波形图和硬件描述语言等方法。

3.3.1 逻辑代数表达式

将逻辑函数的输入变量写成代数表达式，就得到了逻辑代数表达式。例如

$$F(A,B,C,D)=A\overline{B}\,\overline{C}+\overline{A}BC+A\overline{B}\,\overline{D}+\overline{A}\,B\overline{C}D$$

此式表明，输出逻辑变量 F 是输入逻辑变量 A、B、C 和 D 的逻辑函数。它们的逻辑关系由等式右边的代数运算式给出。这种没有括号，以纯粹的与或运算关系表述的方法，也称为乘积项表述，这种表述方法比较适合于计算机处理和自动化设计。

3.3.2 逻辑图表述

逻辑变量之间的函数关系除了可以用代数表达式表述外，还可以用图形符号表述。将

逻辑元件图形符号连接起来表示逻辑函数称为逻辑图。

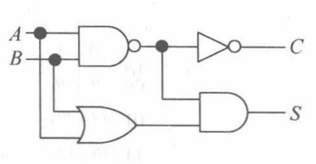


图 3-1 例 3-7 的逻辑图

【例 3-7】分析图 3-1 的逻辑功能。

解：由图 3-1 可知：

$S(A,B)=(A+B)\overline{AB}$　　　　$C(A,B)=\overline{\overline{AB}}=AB$

该逻辑图实现二进制半加器的逻辑功能。S 输出的是 A、B 之和，C 是进位输出位。半加器是没有考虑来自低位电路进位的加法器。

3.3.3 真值表表述

逻辑真值表是一种用表格表示逻辑函数的方法，它是用逻辑变量的所有可能取值组合及其对应的逻辑函数值所构成的表格。真值表的最大优点是能够直接观察出输入与输出之间的逻辑关系。

表 3-2 例 3-8 的真值表

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

【例 3-8】列出函数 $Y=AB+BC+CA$ 的真值表。

解：$Y=AB+BC+CA$ 的真值表如表 3-2 所示。

从真值表 3-2 中可以看出，这是一个多数表决通过的逻辑函数，当输入变量 A、B、C 中有两个或两个以上为 1 时，输出变量 Y 为 1。

事实上，真值表的来源有多种。除了以上直接从逻辑函数获得外，还可以从逻辑电路、逻辑描述（事件）、逻辑控制行为、数字通信方式（规则）、数据采样时序等途径获得。而真值表的获得又是现代数字系统实现自动化设计的一项间接但又十分重要的组成部分，因此应当给予重点关注。

3.3.4 卡诺图表述方式

卡诺图是一种将逻辑函数表示成方格图形的表达形式，它和真值表相似，包含了输入变量的所有可能取值组合以及每种取值组合下的输出结果。但与真值表不同的是，变量取值必须按照循环码的顺序排列。卡诺图与真值表有严格的一一对应的关系。

卡诺图中，方格的数目等于最小项或最大项的总数，即等于 2^n（n 为输入变量数）。所有方格按照格雷码顺序进行行和列的排列，使得每行和每列的相邻方格之间仅有一位变量发生变化。3、4、5 变量的卡诺图分别如图 3-2 (a)、(b)、(c) 所示。

卡诺图是小规模数字电路手工设计的重要工具，是手工逻辑化简的重要途径之一。卡诺图的优点是用几何相邻图形直观地表示了函数的各个最小项在逻辑上的相邻性，以便于对逻辑函数进行化简，用于求逻辑函数的与或最简表达式。

卡诺图的缺点是，只适用于表示和化简逻辑变量较少的逻辑函数，即只适合于小规模逻辑设计中的化简。而现代数字系统设计大多是大规模逻辑设计。此外，由于在数字系统自动设计中，逻辑的化简已经由计算机自动完成，因此已不再使用卡诺图方式来化简逻辑

AB \ C	0	1
00	m_0	m_1
01	m_2	m_3
11	m_6	m_7
10	m_4	m_5

(a)3变量卡诺图

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

(b)4变量卡诺图

AB \ CDE	000	001	011	010	110	111	101	100
00	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
01	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
11	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
10	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

(c)5变量卡诺图

图 3-2　3、4、5 变量的卡诺图

了。但可以通过对卡诺图使用的学习，更加深入理解自动优化设计技术。

3.4　逻辑函数的标准形式

由于逻辑函数的表达形式不是唯一的，为了便于研究，使逻辑函数能与唯一的逻辑函数表达式对应，这里引入逻辑函数标准形式的概念。

逻辑函数的标准形式是建立在最小项和最大项概念的基础之上的。这里首先介绍最小项和最大项的概念，即把逻辑函数转换为标准与或表达式，以及把逻辑函数转换为标准或与表达式。然后讨论两种标准形式的互相转换，以及标准形式与真值表的互相转换。

注意： *逻辑函数的标准表述形式及相关的转换技术也属于数字电路手工设计技术的组成部分，主要目的是有利于真值表表述、卡诺图表述和逻辑化简等，而在自动设计流程中基本不用。*

3.4.1　最小项表述方式

1. 最小项的定义

设有 n 个变量，它们所组成的具有 n 个变量的“与”项中，每个变量以原变量或反变量的形式出现一次，且仅出现一次，则这个乘积项称为最小项。

n 个变量有 2^n 个最小项。例如：4 变量 A、B、C、D 有 16 个最小项为

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D},\quad \overline{A}\,\overline{B}\,\overline{C}D,\quad \overline{A}\,\overline{B}C\overline{D},\quad \cdots,\quad ABCD$$

为了书写方便，把最小项记做 m_i。下标 i 的取值规则是：按照变量顺序将最小项中的原变量用 1 表示，反变量用 0 表示，由此得到一个二进制数。与该二进制数对应的十进制数，即作为下标 i 的值。例如，4 变量 A、B、C、D 的 16 个最小项，可分别记为

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = m_0$$

$$\overline{A}\,\overline{B}\,\overline{C}D = m_1$$

$$\overline{A}\,\overline{B}C\overline{D} = m_2$$

$$\cdots\cdots$$

$$ABCD = m_{15}$$

2. 最小项的性质

(1) 对于任何一个最小项，只有对应的一组变量都取值“1”，才能使其值为“1”。

（2）相同变量构成的两个不同最小项的逻辑与为“0”。

（3）n 个变量的全部最小项之逻辑或为“1”，即 $\sum m_i = 1$。

（4）某一个最小项不是包含在逻辑函数 F 中，就是包含在反函数 $\overline{F}$ 中。

（5）n 个变量构成的最小项有 n 个相邻最小项。相邻最小项是指除一个变量互为相反外，其余变量均相同的最小项。例如，$A\overline{B}CD$ 与 $ABCD$ 是相邻最小项。

3.4.2 最大项表述方式

1. 最大项的定义

设有 n 个变量，它们所组成的具有 n 个变量的“或”项中，每个变量以原变量或反变量的形式出现一次，且仅出现一次，这个“或”项称为最大项。

显然，n 个变量有 2^n 个最大项。例如，4 变量 A、B、C、D 有 16 个最大项：

$$A+B+C+D,\quad A+B+C+\overline{D},\quad A+B+\overline{C}+D,\quad \cdots,\quad \overline{A}+\overline{B}+\overline{C}+\overline{D}$$

为了书写方便，把最大项记做 M_i。下标 i 的取值规则是，按照变量顺序将最大项中的原变量用 0 表示，反变量用 1 表示，由此得到一个二进制数。与该二进制数对应的十进制数即下标 i 的值。例如，4 变量 A、B、C、D 有 16 个最大项，分别可记为

$$A+B+C+D=M_0$$

$$A+B+C+\overline{D}=M_1$$

$$A+B+\overline{C}+D=M_2$$

$$\cdots\cdots$$

$$\overline{A}+\overline{B}+\overline{C}+\overline{D}=M_{15}$$

2. 最大项的性质

（1）对于任何一个最大项，只有对应的一组变量都取值“0”，才能使其值为“0”。其余情况均为“1”，即取值“1”的机会最大。这也就是最大项名字的由来。

例如，只有变量 $ABCD=0000$ 时（每一变量都为 0 时），才有 $A+B+C+D$ 为 0。

（2）相同变量构成的任何两个不同最大项逻辑或为“1”。

例如，在 4 变量最大项中：

$$M_4+M_6=(A+\overline{B}+C+D)+(A+\overline{B}+\overline{C}+D)=A+\overline{B}+1+D=1$$

（3）n 个变量的全部最大项之逻辑与为“0”，即 $\prod M_i = 0$。

（4）某一个最大项不是包含在逻辑函数 F 中，就是包含在反变量 $\overline{F}$ 中。

（5）n 个变量构成的最大项有 n 个相邻最大项。相邻最大项是指除一个变量互为相反外，其余变量均相同的最大项。例如，$A+B+C+D$ 与 $A+B+C+\overline{D}$ 是相邻最大项。

3. 最小项与最大项的关系

下标 i 相同的最小项与最大项互补，即 $m_i=\overline{M_i}$。例如，$\overline{ABC}=\overline{A}+\overline{B}+\overline{C}$，即为 $\overline{m_7}=M_7$。

3.4.3 标准与或表达式

任何一个逻辑函数都可以表示成最小项之和的形式，称为标准与或表达式。如果逻辑函数不是以最小项之和的形式给出，则可以利用公式 $A+\overline{A}=1$ 把它展开成最小项之和的形式。

【例 3-9】将 $F=ABC+\overline{A}B\overline{D}$ 展开为最小项之和的形式。

解：
$$\begin{aligned}F &= ABC+\overline{A}B\overline{D} = ABC(D+\overline{D})+\overline{A}B\overline{D}(C+\overline{C}) \\ &= ABCD+ABC\overline{D}+\overline{A}BC\overline{D}+\overline{A}\,B\overline{C}\,\overline{D} \\ &= m_{15}+m_{14}+m_{6}+m_{4} = \sum m(4,6,14,15)\end{aligned}$$

【例 3-10】将 $F=AB+\overline{A}BC$ 写成标准与或表达式。

解：
$$\begin{aligned}F &= AB+\overline{A}BC = AB(C+\overline{C})+\overline{A}BC \\ &= ABC+AB\,\overline{C}+\overline{A}BC = \sum m(3,6,7)\end{aligned}$$

3.4.4 标准或与表达式

任何一个逻辑函数都可以表示成最大项之积的形式，称为标准或与表达式。如果逻辑函数不是以最大项之积的形式给出，则可利用公式 $A\cdot\overline{A}=0$ 及 $A+BC=(A+B)(A+C)$ 把它展开成最大项之积的形式。

【例 3-11】将 $F=\overline{A}B\overline{C}+AB\overline{C}+\overline{A}BC+\overline{A}\,\overline{B}\,\overline{C}=\sum m(0,2,3,6)$ 展开为最大项之积的形式。

解：
$$\begin{aligned}F &= \overline{\overline{\overline{A}\,B\overline{C}+AB\,\overline{C}+\overline{A}BC+\overline{A}\,\overline{B}\,\overline{C}}} = \overline{\sum m(1,4,5,7)} \\ &= \overline{\overline{A}\,\overline{B}C+A\overline{B}\,\overline{C}+A\overline{B}C+ABC} = \overline{\overline{A}\,\overline{B}C}\cdot\overline{A\overline{B}\,\overline{C}}\cdot\overline{A\overline{B}C}\cdot\overline{ABC} \\ &= (A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+\overline{C}) = \prod M(1,4,5,7)\end{aligned}$$

【例 3-12】将 $F=(A+\overline{B})(A+B+C)$ 写成标准或与表达式。

解：
$$\begin{aligned}F &= (A+\overline{B})(A+B+C) = (A+\overline{B}+C\overline{C})(A+B+C) \\ &= (A+\overline{B}+C)(A+\overline{B}+\overline{C})(A+B+C) = \prod M(0,2,3)\end{aligned}$$

3.4.5 两种标准形式的相互转换

对于一个 n 变量的逻辑函数 F，若 F 的标准与或式由 K 个最小项相或构成，则 F 的标准或与式一定由 2^n-K 个最大项相与构成，并且对于任何一组变量取值组合对应的序号 i，若标准与或式中不含 m_i，则标准或与式中一定含 M_i。

据此，可以根据两种标准形式中的一种直接写出另一种。

【例 3-13】将标准与或表达式 $F(A,B,C)=\sum m(0,3,5,6)$ 表示为标准或与表达式。

解：$F(A,B,C)=\sum m(0,3,5,6)=\prod M(1,2,4,7)$

3.4.6 逻辑函数表达式与真值表的相互转换

1. 由真值表求对应的逻辑函数表达式

如果给出了函数的真值表，则只要将函数值为 1 的那些最小项相加，便是函数的标准与或表达式；将函数值为 0 的那些最大项相乘，便是函数的标准或与表达式。

如果某函数 F 的真值表如表 3-3 所示，则有

$$\begin{aligned} F &= m_1 + m_2 + m_3 + m_5 = \sum m(1,2,3,5) \\ &= \overline{A}\,\overline{B}C + \overline{A}\,B\overline{C} + \overline{A}BC + A\overline{B}C = M_0 \cdot M_4 \cdot M_6 \cdot M_7 = \prod M(0,4,6,7) \\ &= (A+B+C)(\overline{A}+B+C)(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C}) \end{aligned}$$

表 3-3 真值表

A	B	C	F	最小项	最大项
0	0	0	0	m_0	M_0
0	0	1	1	m_1	M_1
0	1	0	1	m_2	M_2
0	1	1	1	m_3	M_3
1	0	0	0	m_4	M_4
1	0	1	1	m_5	M_5
1	1	0	0	m_6	M_6
1	1	1	0	m_7	M_7

2. 由逻辑函数表达式求对应的真值表

由逻辑函数表达式求对应的真值表的步骤是：首先在真值表中列出输入变量二进制值的所有可能取值组合；其次将逻辑函数的与或（或与）表达式转换为标准与或（或与）形式；最后将构成标准与或（或与）形式的每个最小项（最大项）对应的输出变量处填上 1（0），其他填上 0（1）。

例如，逻辑函数 $F=AB+\overline{A}BC=ABC+AB\,\overline{C}+\overline{A}BC$，构成该函数的最小项共有 3 项，即 ABC：111；$AB\,\overline{C}$：110；$\overline{A}BC$：011。则在真值表中，输入变量二进制值 111、110、011 对应的输出变量处填上 1，其他填上 0 即得该函数的真值表。

3.5 逻辑代数化简法

在传统的小规模逻辑设计情况下，逻辑函数的表达式越简单，实现这个逻辑函数所需要的器件就越少，占用硬件资源就越少，逻辑电路结构也就越简单，电路成本也最低。因此，在手工设计流程中，需要通过化简手段找到逻辑函数的最简形式。当然，这种设计理念，在现代数字系统自动设计技术中已不完全正确了。

常用的化简方法有两种，一种是基于逻辑代数的公式化简法，即利用逻辑代数中的公

式和定理进行化简；另一种是图形化简法，即采用卡诺图进行化简。

本节将介绍一些逻辑代数的常用化简方法。

3.5.1　并项化简法

方法是利用公式 $A+\overline{A}=1$，将两项合并成一项，并消去一个变量。

【例 3-14】 化简 $F=A\overline{B}\,\overline{C}+A\overline{B}C$。

解：$F=A\overline{B}\,\overline{C}+A\overline{B}C=A\overline{B}(C+\overline{C})=A\overline{B}$

【例 3-15】 化简 $F=ABC+\overline{A}BC+B\overline{C}$。

解：$F=ABC+\overline{A}BC+B\overline{C}=(A+\overline{A})BC+B\overline{C}=BC+B\overline{C}=B$

【例 3-16】 化简 $F=A(BC+\overline{B}\,\overline{C})+A(B\overline{C}+\overline{B}C)$。

解：$F=A(BC+\overline{B}\,\overline{C})+A(B\overline{C}+\overline{B}C)=A(B\odot C)+A(B\oplus C)$

$=A(B\odot C)+A(\overline{B\odot C})=A$

3.5.2　吸收化简法

吸收化简法是利用公式 $A+AB=A$ 和 $A+\overline{A}B=A+B$，消去多余的项。

【例 3-17】 化简 $F=\overline{A}B+\overline{A}BCD(\overline{E}+F)$。

解：$F=\overline{A}B+\overline{A}BCD(\overline{E}+F)=\overline{A}B$

【例 3-18】 化简 $F=A\overline{B}+C+\overline{A}\,\overline{C}D+B\overline{C}D$。

解：$F=A\overline{B}+C+\overline{A}\,\overline{C}D+B\overline{C}D=A\overline{B}+C+\overline{C}(\overline{A}+B)D$

$=A\overline{B}+C+(\overline{A}+B)D=A\overline{B}+C+\overline{A\overline{B}}D$

$=A\overline{B}+C+D$

【例 3-19】 化简 $F=A+\overline{\overline{A}\,\overline{BC}}(\overline{A}+\overline{\overline{B}\,\overline{C}+D})+BC$。

解：$F=A+\overline{\overline{A}\,\overline{BC}}(\overline{A}+\overline{\overline{B}\,\overline{C}+D})+BC=A+BC+(A+BC)(\overline{A}+\overline{\overline{B}\,\overline{C}+D})=A+BC$

3.5.3　配项化简法

方法一，利用公式 $A+\overline{A}=1$，给某一个与项配项，然后将其拆分成两项，再与其他项合并。

【例 3-20】 化简 $F=AB+\overline{A}\,\overline{C}+\overline{B}\,\overline{C}$。

解：$F=AB+\overline{A}\,\overline{C}+\overline{B}\,\overline{C}=AB+\overline{A}\,\overline{C}+A\overline{B}\,\overline{C}+\overline{A}\,\overline{B}\,\overline{C}$

$=AB+\overline{A}\,\overline{C}+A\overline{B}\,\overline{C}=AB+\overline{C}(\overline{A}+A\overline{B})$

$=AB+\overline{C}(\overline{A}+\overline{B})=AB+\overline{C}\,\overline{AB}=AB+\overline{C}$

【例 3-21】 化简 $F=\overline{A}\,\overline{B}+\overline{B}\,\overline{C}+BC+AB$。

解：$F=\overline{A}\,\overline{B}+\overline{B}\,\overline{C}+BC+AB=\overline{A}\,\overline{B}(C+\overline{C})+\overline{B}\,\overline{C}+BC(A+\overline{A})+AB$

$=\overline{A}\,\overline{B}C+\overline{A}\,\overline{B}\,\overline{C}+\overline{B}\,\overline{C}+ABC+\overline{A}BC+AB$

$$=AB+\overline{B}\,\overline{C}+\overline{A}C(B+\overline{B})=AB+\overline{B}\,\overline{C}+\overline{A}C$$

方法二，利用公式 $A+A=A$，为某项配上其所能合并的项。

【例 3-22】 化简 $F=ABC+AB\overline{C}+A\overline{B}C+\overline{A}BC$。

解：$F=ABC+AB\overline{C}+A\overline{B}C+\overline{A}BC$

$=(ABC+AB\overline{C})+(ABC+A\overline{B}C)+(ABC+\overline{A}BC)$

$=AB+AC+BC$

3.5.4 消去冗余项化简法

此方法是利用公式 $AB+\overline{A}C+BC=AB+\overline{A}C$，将冗余项 BC 消去。且含冗余项 BC 的“与”项仍是冗余项，如：ABC。

【例 3-23】 化简 $F=AC+A\overline{B}CD+ABC+\overline{C}D+ABD$。

解：$F=AC+A\overline{B}CD+ABC+\overline{C}D+ABD$

$=AC(1+\overline{B}D+B)+\overline{C}D+ABD$

$=AC+\overline{C}D+ABD=AC+\overline{C}D$

【例 3-24】 化简 $F=AB+\overline{B}C+AC(DE+FG)$。

解：$F=AB+\overline{B}C+AC(DE+FG)=AB+\overline{B}C$

注意： 实际应用中遇到的逻辑函数往往比较复杂，化简时应灵活使用所学的定律、公式及规则，综合运用各种方法。

【例 3-25】 化简 $F=AD+A\overline{D}+AB+\overline{A}C+BD+\overline{B}E+DE$。

解：$F=AD+A\overline{D}+AB+\overline{A}C+BD+\overline{B}E+DE$

$=A+AB+\overline{A}C+BD+\overline{B}E+DE$

$=A+\overline{A}C+BD+\overline{B}E+DE$

$=A+C+BD+\overline{B}E+DE$

$=A+C+BD+\overline{B}E$

【例 3-26】 化简 $F=AB+A\overline{C}+\overline{B}C+\overline{C}B+\overline{B}D+\overline{D}B+ADE(F+G)$。

解：$F=AB+A\overline{C}+\overline{B}C+\overline{C}B+\overline{B}D+\overline{D}B+ADE(F+G)$

$=A(B+\overline{C})+\overline{B}C+\overline{C}B+\overline{B}D+\overline{D}B+ADE(F+G)$

$=A(\overline{\overline{B}C})+\overline{B}C+\overline{C}B+\overline{B}D+\overline{D}B+ADE(F+G)$

$=A+\overline{B}C+\overline{C}B+\overline{B}D+\overline{D}B+ADE(F+G)$

$=A+\overline{B}C(D+\overline{D})+\overline{C}B+\overline{B}D+\overline{D}B(C+\overline{C})$

$=A+\overline{B}CD+\overline{B}C\overline{D}+\overline{C}B+\overline{B}D+\overline{D}BC+\overline{D}B\overline{C}$

$=A+\overline{B}D+C\overline{D}+B\overline{C}$

【例 3-27】 化简 $F=(\overline{B}+D)(\overline{B}+D+A+G)(C+E)(\overline{C}+G)(A+E+G)$。

解：(1) 先求出 F 的对偶函数 F^*，并对其进行化简：

$$F^*=\overline{B}D+\overline{B}DAG+CE+\overline{C}G+AEG=\overline{B}D+CE+\overline{C}G$$

(2) 求 F^* 的对偶函数，便得 F 的最简或与表达式：

$$F=(\overline{B}+D)(C+E)(\overline{C}+G)$$

3.6 卡诺图化简法

逻辑代数化简法的缺点是除了需要掌握一定的化简技巧外，还必须能熟练运用逻辑代数的公式、定理和定律。相比而言，较简单易行的方法是使用卡诺图来化简。卡诺图化简法是由美国工程师卡诺提出来的，它是一种图形化简方法。本节介绍 2 变量、3 变量和 4 变量卡诺图、与或表达式的卡诺图表达方式、与或表达式的卡诺图和或与表达式的卡诺图的化简方法，以及含无关项逻辑函数、多输出逻辑函数的卡诺图化简等。

3.6.1 与或表达式的卡诺图表示

对于标准形式的与或表达式来说，卡诺图的表示方法是，把表达式中的每一个最小项所对应的方格中填入 1，其余方格填入 0，就得到了该逻辑函数的卡诺图。

【例 3-28】 用卡诺图表示标准与或表达式：$F=\overline{A}BC+ABC+A\overline{B}C$。

解：绘出的卡诺图如图 3-3 所示。如果是非标准形式的与或表达式，可先利用 3.4 节所介绍的方法将它转换为标准形式。

【例 3-29】 用卡诺图表示逻辑函数：$F=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}B\overline{D}+ACD+A\overline{B}$。

解：首先将逻辑函数 F 化为若干个最小项之和的标准形式：

$$
\begin{aligned}
F&=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}B\overline{D}+ACD+A\overline{B}\\
&=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}B(C+\overline{C})\overline{D}+A(B+\overline{B})CD+A\overline{B}(C+\overline{C})(D+\overline{D})\\
&=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}B\,C\overline{D}+\overline{A}\,B\overline{C}\overline{D}+ABCD+A\overline{B}CD+A\overline{B}\,C\overline{D}+A\overline{B}\,\overline{C}D+A\overline{B}\,\overline{C}\,\overline{D}\\
&=\sum m(1,5,6,8,9,10,11,15)
\end{aligned}
$$

然后根据获得的标准形式，画出 4 变量的卡诺图。即在对应于该函数 F 中各最小项的方格中填入 1，其余方格中填入 0，就得到了如图 3-4 所示的卡诺图。

当然，也可直接由非标准形式的与或表达式得到相应的卡诺图，详见下例。

【例 3-30】 用卡诺图表示逻辑函数：$F=\overline{A}\,\overline{D}+\overline{B}C$。

解：在变量 A、D 取值均为 00 的所有方格中填入 1；在变量 B、C 取值分别为 0、1 的所有方格中填入 1，其余方格中填入 0，卡诺图如图 3-5 所示。

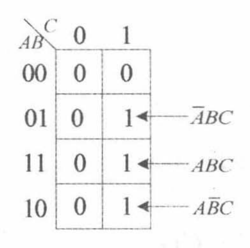


图 3-3 标准与或卡诺图

AB \ CD	00	01	11	10
00	0	1	0	0
01	0	1	0	1
11	0	0	1	0
10	1	1	1	1

图 3-4 标准与或表达式卡诺图

AB \ CD	00	01	11	10
00	1	0	1	1
01	1	0	0	1
11	0	0	0	0
10	0	0	1	1

图 3-5 非标准与或表达式卡诺图

3.6.2 与或表达式的卡诺图化简

1. 卡诺图化简原理

在卡诺图中，如果两个最小项之间只有一个变量取值不同，其余变量相同，则称它们具有“逻辑相邻性”，这两个最小项称为“逻辑相邻最小项”，如 ABC 和 $AB\overline{C}$。

显然，几何位置相邻（仅指上下或左右，不包括对角）的方格所对应的最小项一定是逻辑相邻最小项。此外，与卡诺图中心轴对称的左右两边方格对应的最小项之间，及上下两边方格对应的最小项之间，也是逻辑相邻最小项。例如，图 3-6 中 m_0 与 m_2 之间，及 m_1 与 m_9 之间也是逻辑相邻最小项。

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

图 3-6 逻辑相邻最小项的概念

两个相邻最小项可以合并为一项，并消去一个变量，例如：$AB+A\overline{B}=A$。

这就是卡诺图化简的依据。因此，逻辑函数化简的实质就是在卡诺图中寻找逻辑相邻最小项，并将它们进行合并。

2. 卡诺图化简的步骤

步骤 1： 对卡诺图中的 1 进行分组，并将每组用“圈”围起来。根据以下规则分组：

（1）每个圈内只能含有 2^n（$n=0$，1，2，3，…）个最小项。

（2）圈内的每一个最小项必须和该圈中的一个或多个最小项逻辑相邻，但该圈中的所有最小项并不一定必须相互逻辑相邻。

（3）所有取值为 1 的方格均要被圈过，即不能漏下取值为 1 的方格，但它们可以多次被圈；

（4）圈的个数尽量少，圈内方格的个数尽可能多。

步骤 2： 由每个圈得到一个合并的与项。该与项由该圈中仅仅以一种形式（原变量或者反变量）出现的所有变量构成。即消去同时以原变量和反变量形式出现的变量。

步骤 3： 将上一步各合并与项相加，即得所求的最简与或表达式。

【例 3-31】 用卡诺图化简法求出以下逻辑函数的最简与或式：

$$F(A,B,C,D)=\sum m(1,3,5,7,8,9,10,11,14,15)$$

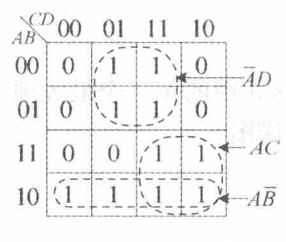


图 3-7 例 3-31 的卡诺图

解：首先画出该逻辑函数的卡诺图（图 3-7），然后按照以上步骤进行化简。

该卡诺图中的“1”被分为 3 组，分别对应 3 个圈。第一个圈中出现了 B 和 $\overline{B}$、C 和 $\overline{C}$，所以这些变量被消去。留下了变量 $\overline{A}$ 和 D 而形成合并后的与项 $\overline{A}D$。

同样方法可得到其他两个圈对应的与项 AC 和$A\overline{B}$。最后将这 3 个与项相加（相或）就得到如下最简与或表达式：

$$F(A,B,C,D)=\overline{A}D+AC+A\overline{B}$$

【例 3-32】 某逻辑电路的输入变量为 A、B、C、D，它的真值表如表 3-4 所示，用卡诺图化简法求出逻辑函数 $F(A,\ B,\ C,\ D)$ 的最简与或表达式。

表 3-4　例 3-32 的真值表

A	B	C	D	F	A	B	C	D	F
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	1	1	1	0	0	1
0	1	0	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

解：由以上真值表画出卡诺图，如图 3-8 所示。找出可以合并的最小项，即画“圈”，并写出最简与或表达式：

$$F=\overline{C}\,\overline{D}+A\overline{B}\,\overline{D}+\overline{A}B\overline{C}+ABCD$$

【例 3-33】 用卡诺图化简法求出以下逻辑函数的最简与或式：

$$F(A,B,C,D)=\sum m(0,2,3,4,6,8,10,11,12,14)$$

解：画出该逻辑函数的卡诺图（图 3-9）。最简与或式为

$$F(A,B,C,D)=\overline{B}C+\overline{D}$$

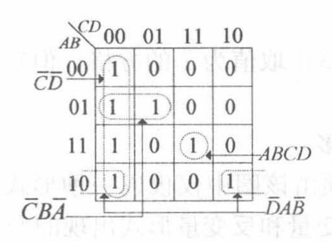


图 3-8　例 3-32 的卡诺图

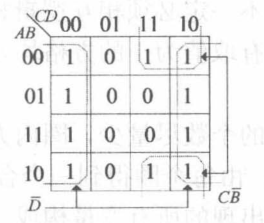


图 3-9　例 3-33 的卡诺图

3.6.3　或与表达式的卡诺图化简

1. 或与表达式的卡诺图表示

对于标准形式的或与表达式来说，卡诺图的表示方法是：把表达式中的每一个最大项所对应的方格中填入 0，其余方格填入 1，就得到了该逻辑函数的卡诺图。

对于非标准形式的或与表达式，可将其变换为标准形式。

【例 3-34】 用卡诺图表示下面的标准或与表达式：

$$F=(A+B+C)(A+\overline{B}+C)(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})$$

解：根据上式画出卡诺图，如图 3-10 所示。

2. 或与表达式的卡诺图化简

或与表达式的卡诺图化简过程和与或表达式的卡诺图化简过程基本上是一样的，只不过对卡诺图中的“0”进行分组而产生合并的或项，将各合并的或项相与，即得所求的最简或与表达式。

【例 3-35】用卡诺图化简下面或与表达式：

$$F=(A+B+C)(A+\overline{B}+C)(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})$$

解：根据上式画出卡诺图，如图 3-11 所示。

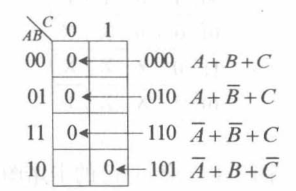


图 3-10 标准或与表达式的卡诺图

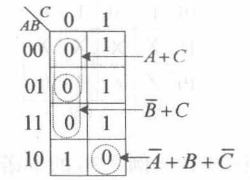


图 3-11 例 3-34 的卡诺图

将以上卡诺图中的“0”分为 3 组，分别对应 3 个圈。第 1 个圈中出现了 B 和 $\overline{B}$，所以这个变量被消去。留下了变量 A 和 C，形成合并后的或项 $A+C$。同样方法可得到其他两个圈对应的或项 $\overline{B}+C$ 和 $\overline{A}+B+\overline{C}$。将这 3 个或项相与，就得到最简或与表达式：

$$F=(A+C)(\overline{B}+C)(\overline{A}+B+\overline{C})$$

3.6.4 含无关项逻辑函数的化简

在进行逻辑设计时，有时会遇到这种情况，对于输入变量的某些取值，输出函数值可以是任意的，或者这些变量的取值组合根本不会出现。这些变量的取值组合所对应的最小项称为“任意项”。用符号“d”、“×”或“ϕ”表示，其函数值可以是 0 或 1。

使用无关项有助于逻辑函数的化简。含无关项的逻辑函数可用下式表示。

最小项表达式： $F=\sum m()+d()$ 或者 $\begin{cases}F=\sum m()\\ \sum d()=0\end{cases}$

【例 3-36】化简下列函数：

$$F(A,B,C,D)=\sum m(0,3,4,7,11)+d(8,9,12,13,14,15)$$

解：上式中对应于最小项 m_0、m_3、m_4、m_7、m_{11} 的方格中填入 1，而对应于无关项 m_8、m_9、m_{12}、m_{13}、m_{14}、m_{15} 的方格中填入×，表示其取值不确定，其余位置填入 0。卡诺图如图 3-12 所示。

在化简过程中，无关项的取值可视具体情况取 0 或取 1。原则是如果无关项对化简有利，则取 1；如果无关项对化简不利，则取 0。

当无关项 m_8、m_{12},、m_{15} 均取 1，其他无关项 m_9、m_{13},、m_{14}，均取 0 时，函数 F 可化简为：$F=\overline{C}\,\overline{D}+CD$。

【例 3-37】 化简函数：$F=\overline{B}C\overline{D}+\overline{A}\,\overline{B}CD+A\overline{B}\,\overline{D}+\overline{B}\,\overline{C}\,\overline{D}$。已知约束条件为 $AD+BC=0$。

解：将上述约束条件变换为最小项之和的形式：

$$AD+BC=A(B+\overline{B})(C+\overline{C})D+(A+\overline{A})BC(D+\overline{D})$$
$$=ABCD+AB\overline{C}D+A\overline{B}CD+A\overline{B}\,\overline{C}D+\overline{A}BCD+ABC\overline{D}+\overline{A}BC\overline{D}=0$$

以上等式的 0 左边的最小项都是无关项，即 $d(6,7,9,11,13,14,15)=0$，由此可得到逻辑函数 F 的卡诺图如图 3-13 所示。

AB\CD	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	X	X	1	0
	$\overline{C}\,\overline{D}$		CD	

图 3-12 例 3-36 的卡诺图

AB\CD	00	01	11	10
00	1	0	1	1
01	0	0	X	X
11	0	X	X	X
10	1	X	X	1

图 3-13 例 3-37 的卡诺图

由此卡诺图可得最简与或表达式为：$F=C+\overline{B}\,\overline{D}$。

3.6.5 多输出逻辑函数的化简

在实际逻辑问题中，大量存在着由同一组输入变量产生多个输出函数的问题，实现这类问题会涉及多输出逻辑函数的化简。

在进行多输出逻辑函数的化简时，如果只是孤立地求出各输出逻辑函数的最简表达式，然后设计出相应的逻辑电路图，并将其拼在一起，一般不能保证逻辑电路整体最简。因为各输出函数之间往往存在相互联系，具有某些共同的部分。因此，应该将它们当作一个整体来考虑，而不应该将其完全分开。使这类逻辑电路达到最简的关键，在于函数化简时找出各输出函数的公用项，以便在逻辑电路中实现对公用项逻辑部件的共享，从而使电路整体最简。

AB\CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	1
10	0	0	1	1

(a) F_1 的卡诺图

AB\CD	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	1	1	0	1
10	0	0	0	1

(b) F_2 的卡诺图

图 3-14 例 3-38 的卡诺图

【例 3-38】 化简以下多输出函数：

$$F_1=\sum m(2,3,6,7,10,11,12,13,14,15)$$
$$F_2=\sum m(2,6,10,12,13,14)$$

解：分别作出它们的卡诺图，如图 3-14 所示。观察两个卡诺图，找出两者相同的部分，并化简为：$F_1=C+AB\overline{C}$；$F_2=C\overline{D}+AB\overline{C}$。

虽然 F_1 不是最简，但整体达到了最简。

习 题

3-1 用真值表证明下列等式：

(1) $AB+\overline{A}C+BC=AB+\overline{A}C$

(2) $AB+BC+CA=(A+B)(B+C)(C+A)$

3-2 用逻辑代数证明下列各等式：

(1) $A+\overline{A}C+CD=A+C$

(2) $(A+B)(\overline{A}+C)(B+C)=(A+B)(\overline{A}+C)$

(3) $A\oplus\overline{B}=\overline{A}\,\overline{B}+AB$

(4) $(A+B)(A+B+C+D+E+F)=A+B$

(5) $BC+D+\overline{D}(\overline{B}+\overline{C})(AD+B)=B+D$

3-3 用公式法化简下列逻辑函数为最简与或表达式：

(1) $Y_2=A+ABC+A\overline{B}\,\overline{C}+BC+\overline{B}\,\overline{C}$

(2) $Y_3=\overline{A}\,\overline{B}(C+CD)+A\overline{B}D$

(3) $Y_4=A\overline{B}+\overline{A}CD+B+\overline{C}+D$

(4) $Y_5=\overline{\overline{\overline{A}C\overline{C}\,\overline{D}}\;\overline{\overline{B}D}}+BC+\overline{\overline{A}\,\overline{\overline{C}\,\overline{D}}}+\overline{A}+\overline{\overline{B}D}$

3-4 画出下列各函数用与非运算表示的逻辑图（即只用与非门构建的电路实现以下逻辑函数）：

(1) $Y=A\overline{B}+\overline{A}C$

(2) $Y=\overline{A\overline{B}+\overline{A}\,B}$

(3) $Y=\overline{AB}D+\overline{A}BC+ABD+A\overline{B}C$

3-5 写出图 3-15 各逻辑图所对应的逻辑函数式，列出它们的真值表。

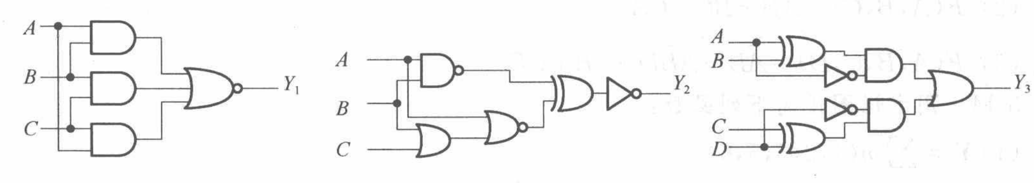


图 3-15 题 3-5 的逻辑图

3-6 试分析图 3-16 所示逻辑电路的功能。

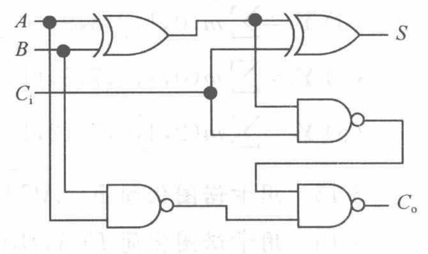


图 3-16 题 3-6 逻辑电路

3-7 写出下列函数 F 的对偶函数 F^* 及反函数 $\overline{F}$：

(1) $F=\overline{A}\,\overline{C}+BD$

(2) $F=\overline{A+\overline{B+C}}$

(3) $F=\overline{\overline{AD}+ACD}\cdot(BC+D)$

(4) $F=(A+B)\cdot(\overline{B}+C)+AD+\overline{E}$

3-8 用代数法化简下列逻辑式：

(1) $A+ABC+A\,\overline{BC}+BC+\overline{B}C$

(2) $\overline{AB+\overline{A}\,\overline{B}+\overline{A}B+A\overline{B}}$

(3) $\overline{\overline{(\overline{A}+B)}+\overline{(A+B)}+\overline{(\overline{A}B)}\,\overline{(A\overline{B})}}$

(4) $(A+B+\overline{C})(A+B+C)$

(5) $\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}C+ABC+A+B\overline{C}$

(6) $\overline{B}+ABC+\overline{A}\,\overline{C}+\overline{A}\,\overline{B}$

(7) $\overline{\overline{AC+\overline{A}BC}+\overline{B}C+AB\overline{C}}$

(8) $\overline{\overline{A\overline{B}+ABC}+A(B+A\overline{B})}$

(9) $ABC\overline{D}+ABD+BC\overline{D}+ABCD+B\overline{C}$

3-9 试画出逻辑函数 $F(A,B,C,D)=(A+C)\overline{(\overline{A}+\overline{C})}$ 的卡诺图。

3-10 将下列函数展开成最大项表达式：

(1) $F=A\oplus B+\overline{AC}$

(2) $F=(\overline{A}\oplus B)+A(B\oplus C)$

3-11 用卡诺图化简下列用最大项表示的函数：

(1) $F(A,B,C,D)=\prod M(0,1,4,5,9,13)$

(2) $F(A,B,C,D)=\prod M(4,5,6,7,9,12)$

3-12 用卡诺图化简下列非标准形式的逻辑函数：

(1) $Y=\overline{A}\,\overline{B}\,\overline{C}+AC+A\overline{B}\,\overline{C}+B\overline{C}$

(2) $Y=(\overline{A}+\overline{C})D+\overline{C}(B+AD)$

3-13 将下列函数展开为最小项表达式：

(1) $F(A,B,C)=\overline{A(\overline{B}+C)}$

(2) $F(A,B,C)=AB+BC+CA$

(3) $F(A,B,C,D)=\overline{\overline{A}B+A\overline{B}D}\cdot(\overline{B}+CD)$

3-14 用卡诺图化简下列函数：

(1) $Y_1=\sum m(1,2,3,5,6)$

(2) $Y_2=\sum m(2,3,4,5,8,9,14,15)$

(3) $Y_3=\sum m(0,1,2,8,9)+\sum d(10,11,12,13,14,15)$

(4) $Y_4=\sum m(0,1,5,7,8,11,14)+\sum d(3,9,15)$

(5) $Y_5=\sum m(2,4,6,7,12,15)+\sum d(0,1,3,8,9,11)$

3-15 用卡诺图化简 $F=\overline{AC+\overline{A}BC+\overline{B}C}+AB\overline{C}$，并利用或非门实现函数功能。

3-16 用卡诺图化简 $F(A,B,C,D)=\sum m(0,2,8,10,14,15)$，并利用与非门实现函数功能。

3-17 化简下列函数，并利用或非门实现函数功能。

(1) $F(A,B,C)=\sum m(0,2,3,7)$

(2) $F(A,B,C,D)=\sum m(0,1,2,4,6,10,14,15)$

第4章 组合逻辑电路的分析与设计

本章将以多个不同类型的示例，给出基于手工设计技术的组合逻辑电路的分析及设计方法，然后详细介绍一些经典的组合逻辑电路模块的逻辑功能和使用方法，其中包括编码器、译码器、数据选择器、加法器、数据比较器等，为读者深入了解实用的数字逻辑电路作必要的铺垫。

最后，为了能在第6章（作为本章内容的延续）中完整地介绍基于现代数字系统自动设计的数字电路设计与分析技术，首先将本章给出的不同类型的组合电路功能模块作一归纳，引入了广义译码器的概念，然后介绍可编程逻辑器件的基本结构、工作原理及其与自动设计技术的关系。

4.1 组合逻辑电路手工分析

按照逻辑功能的不同特点，可以把数字电路分成两大类，一类叫做组合逻辑电路，另一类叫做时序逻辑电路。本节首先介绍组合逻辑电路的基本概念、特点和表示方法，然后介绍针对组合逻辑电路的手工分析数字技术。

所谓手工数字技术就是无需任何形式的计算机的参与而纯粹通过手工或者说人工的努力完成的数字电路分析和设计的技术，这是传统数字电路技术的主要内容。

4.1.1 组合逻辑电路的定义

组合逻辑电路是将逻辑门以一定的方式组合在一起，使其具有一定逻辑功能的电路。组合逻辑电路是一种无记忆电路；即在任何时刻，电路的输出状态仅取决于这一时刻的输入状态（假设不考虑延时情况），而与电路原来的（当前时刻以前的）状态无关，这样的逻辑电路称为组合逻辑电路，简称为组合电路。组合电路结构如图4-1所示。

组合逻辑电路的输出与输入之间可以用如下逻辑函数表示：

$$z_i = f_i(x_1, x_2, \cdots, x_n) \quad (i = 1, 2, \cdots, m)$$

由以上讨论可知，组合电路有两个主要特点：一是由逻辑门电路组成；二是输出与输入之间不存在反馈回路。

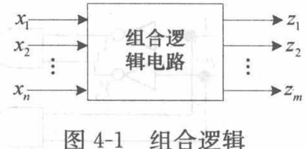


图4-1 组合逻辑电路框图

组合电路可以通过逻辑函数表达式、真值表、卡诺图、逻辑图及波形图来表述和加以分析，这几种表述方法在电路分析中各自具有不同的特点：

- 逻辑表达式。一般为与或式，但形式不唯一，通过变换可实现用不同门电路组成

逻辑图。在一定程度上可以直接用于自动设计的描述，如转化为硬件描述语言的描述。

- 真值表。直观地反映出变量取值与函数值之间的关系，通过对其进行状态赋值可以得到对应的真值表。真值表是判断逻辑关系的有效手段，真值表具有唯一性（表述方法不唯一）。在自动设计中，用真值表描述逻辑有利于计算机对逻辑的自动设计。
- 卡诺图。是化简逻辑函数的主要工具，为最后实现逻辑图作必要准备。
- 逻辑图。表示变量之间的逻辑关系，一个逻辑表达式可用不同逻辑图实现。逻辑图只反映逻辑功能，不反映电路的时序特性。逻辑图只适合表达局部或小规模逻辑。

4.1.2 组合逻辑电路的手工分析步骤

组合逻辑电路分析的目的就是根据给定的逻辑电路获得与之对应的逻辑功能。手工分析步骤如下：

(1) 根据给定的逻辑电路，写出输出逻辑函数表达式。
(2) 用卡诺图或公式法化简逻辑函数表达式。
(3) 列出输入输出关系真值表。
(4) 根据真值表说明电路的逻辑功能。

4.1.3 组合逻辑电路分析实例

以下给出两则示例来具体说明手工分析组合电路的方法。

1. 单输出组合逻辑电路的分析

【例 4-1】 已知逻辑电路如图 4-2 所示，分析该电路逻辑功能。

解：(1) 根据给定的逻辑电路，逐级写出各输出的逻辑函数表达式：

$$Y_1 = \overline{A} \qquad Y_2 = \overline{B}$$

$$Y_3 = Y_1 \cdot Y_2 = \overline{A} \cdot \overline{B} \qquad Y_4 = A \cdot B$$

$$Y = Y_3 + Y_4$$

(2) 化简逻辑电路的输出函数表达式：

$$Y = Y_3 + Y_4 = \overline{A} \cdot \overline{B} + A \cdot B = A \odot B$$

(3) 根据化简后的逻辑函数表达式列出真值表，如表 4-1 所示。

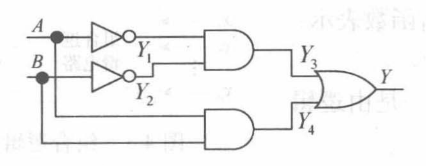


图 4-2 单输出组合逻辑电路图

表 4-1 例 4-1 真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(4) 该电路实现的是同或逻辑功能。

2. 多输出组合逻辑电路的分析

【例 4-2】 已知逻辑电路如图 4-3 所示，分析该电路的逻辑功能。

解：(1) 根据给定的逻辑电路，写出所有输出逻辑函数表达式，并对其进行化简。

$$L_1 = A \cdot \overline{B} \qquad L_3 = \overline{A} \cdot B$$

$$L_2 = \overline{A \cdot \overline{B} + \overline{A} \cdot B} = \overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B} = (\overline{A} + B) \cdot (A + \overline{B}) = A \cdot B + \overline{A} \cdot \overline{B} = A \odot B$$

(2) 根据化简后的逻辑函数表达式列出真值表，如表 4-2 所示。

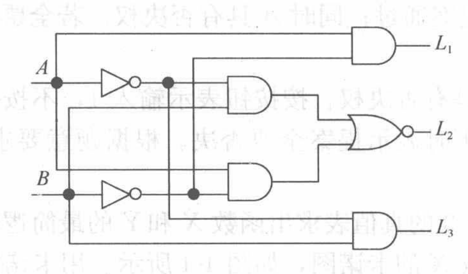


图 4-3 多输出组合逻辑电路图

表 4-2 例 4-2 真值表

A	B	L_1	L_2	L_3
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

(3) 逻辑功能说明。该电路是一位二进制数比较器，当 $A=B$ 时，$L_2=1$（其余输出为 0）；当 $A>B$ 时，$L_1=1$（其余输出为 0）；当 $A<B$ 时，$L_3=1$（其余输出为 0）。

4.2 组合逻辑电路手工设计方法

这里主要介绍组合逻辑电路的手工设计步骤。通过几个组合逻辑电路的设计实例，给出一般设计技术，然后给出用小规模集成电路实现组合电路逻辑功能的方法。

4.2.1 组合逻辑电路的一般设计步骤

在进行组合逻辑电路的设计时，需要根据给定的实际逻辑问题，设计出一个含最简结构的逻辑电路图。设计组合逻辑电路应该遵循的主要步骤如下：

(1) 对实际逻辑问题进行逻辑抽象，确定输入、输出变量；分别对输入、输出变量的具体含义进行定义，然后根据输出与输入之间的逻辑关系列出真值表。

(2) 根据真值表写出相应的逻辑函数表达式。

(3) 将逻辑函数表达式化简，并转换成所需要的形式。

(4) 根据最简逻辑函数表达式画出逻辑电路图。

事实上，组合逻辑电路的设计流程恰好是 4.1.2 节给出的分析流程的逆过程。这里给出的设计步骤仅是基于传统的手工设计步骤，至于现代的自动设计步骤，则要方便多了。在自动设计流程中，主要的人工设计的工作只需完成以上第 (1) 步即可，余下的工作都可由计算机来完成，详细方法将在第 6 章中介绍。

4.2.2 组合逻辑电路的设计示例

实用数字电路系统中常遇到的编码器、译码器、多路选择器、加法器、奇偶校验器、比较器和部分只读存储器等，都属于组合逻辑电路。

下面以编码器、译码器、比较器等典型组合电路为例，具体地说明组合电路的特点、分析和设计方法。

【例 4-3】 分别用与非门和或非门设计一个表决电路。即设计一个 A、B 和 C 共 3 人的表决电路。当表决某个提案时，多数人同意，则提案通过；同时 A 具有否决权。若全票否决，也给出显示。

解：(1) 进行逻辑抽象，建立真值表。设 A 具有否决权。按按钮表示输入 1，不按按钮表示输入 0；以 X 为 1 时表示提案通过；Y 为 1 时表示提案全票否决。根据题意要求，列出真值表如表 4-3 所示。

表 4-3 例 4-3 真值表

A	B	C	X	Y
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

(2) 根据表 4-3 的真值表求出函数 X 和 Y 的最简逻辑表达式。画出函数 X 的卡诺图，如图 4-4 所示。用卡诺图化简后得到函数的最简与或表达式为

$$X = AB + AC$$

实现逻辑函数的电路图如图 4-5 (a) 所示。

(3) 题目要求使用与非门，故将上述表达式变换成与非表达式：

$$X = AB + AC = \overline{\overline{AB + AC}} = \overline{\overline{AB} \cdot \overline{AC}}$$

(4) 用与非门画出实现上述逻辑表达式的逻辑电路图如图 4-5 (b) 所示。

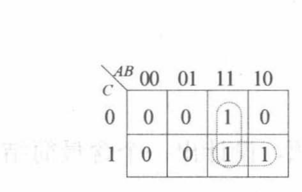
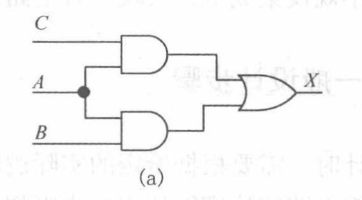


图 4-4 例 4-3 函数 X 的卡诺图



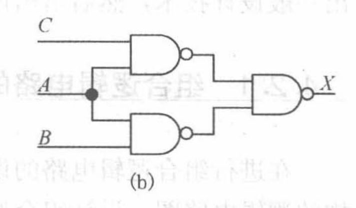


图 4-5 例 4-3 的逻辑电路图

(5) 观察表 4-3 可以直接获得 Y 的逻辑表述如下。这可用一个 3 输入或非门实现。

$$Y = \overline{A}\,\overline{B}\,\overline{C} = \overline{A + B + C}$$

4.3 编 码 器

本节首先介绍编码器的基本概念，然后介绍一些不同类型的实用编码器的功能、使用方法及设计方法，其中包括二进制编码器、优先编码器和二-十进制编码器。

4.3.1 编码器的基本概念

在数字系统中，通常将具有特定含义的信息（数字或符号）编成相应的若干位二进制代码的过程，称为编码。实现编码功能的电路称为编码器（Encoder）。其功能框图如图 4-6 所示。编码器的特点是当多个输入端的其中一个为有效电平时，编码器的输出端并行输出相应的多位二进制代码。按照被编码信号的不同特点和要求，有二进制编码器、BCD 码编码器、优先编码器等多种形式。



图 4-6 编码器框图

4.3.2 二进制编码器

用 n 位二进制代码对 $M=2^n$ 个信号进行编码的电路叫二进制编码器。以下介绍两种常用的二进制编码器。

1. 3 位二进制编码器

3 位二进制编码器的框图如图 4-7 所示。它有 8 个数据输入端，分别代表 8 种需要编码的信息，用 $I_0 \sim I_7$ 表示。输出端是用来进行编码的 3 位二进制代码（即构成输入端信息对应的编码输出），用 C、B、A（习惯上设 C 为最高位，A 为最低位）来表示。由于编码器在任何时刻只能对一个输入端信号进行编码，所以不允许两个或两个以上输入端同时存在有效信号。如对 I_0 进行编码，就是使输入端 I_0 有效，而其他输入端无效，此时输出有一组代码相对应。有效信号有两种表达方式：一是 I_0 加高电平而其他输入端加低电平，这称为“输入高电平有效”；另一种相反，称为“输入低电平有效”。



图 4-7 3 位二进制编码器框图

这里设输入端信号高电平有效，对输入信号 $I_0 \sim I_7$ 进行编码：C、B、A 为编码器输出，则可得到如表 4-4 所示的真值表。由于 $I_0 \sim I_7$ 是一组互相排斥的变量（编码），所以真值表可以采用简化形式列出来，形式如表 4-5 所示。

表 4-4 3 位二进制编码器的真值表

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	C	B	A
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

表 4-5 简化真值表

输 入	输 出		
I	C	B	A
I_0	0	0	0
I_1	0	0	1
I_2	0	1	0
I_3	0	1	1
I_4	1	0	0
I_5	1	0	1
I_6	1	1	0
I_7	1	1	1

表 4-5 显示，只需要将输出端为 1 的变量相或，便可得到输出端的最简与或表达式：

$$C = I_4 + I_5 + I_6 + I_7 = \overline{\overline{I_4 + I_5 + I_6 + I_7}} = \overline{\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7}$$

$$B = I_2 + I_3 + I_6 + I_7$$

$$A = I_1 + I_3 + I_5 + I_7$$

根据上述各表达式可直接画出 3 位二进制编码器的逻辑电路图，如图 4-8 所示。图 4-9 所示的是 74 系列中常用的一种标准功能专用集成电路的编码器逻辑符号。

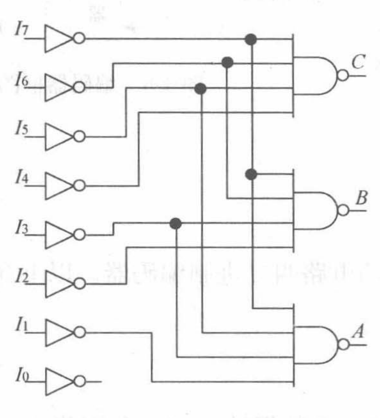


图 4-8　3 位二进制编码器电路图

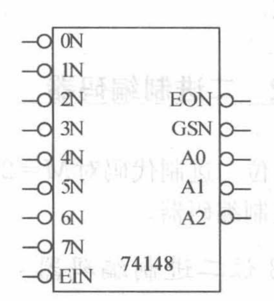


图 4-9　8-3 优先编码器逻辑符号

2. 优先编码器

优先编码器的功能是事先对输入端进行优先级别排序，在任何时刻仅对优先级别高的输入端信号响应，优先级别低的输入端信号则不响应。

表 4-6 是 8-3 线优先编码器 74LS148 的真值表，图 4-9 是对应的元件逻辑符号。

表 4-6　74LS148 真值表

输　入									输　出				
EIN	7N	6N	5N	4N	3N	2N	1N	0N	GSN	EON	A_2	A_1	A_0
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	×	×	×	×	×	×	×	0	1	0	0	0
0	1	0	×	×	×	×	×	×	0	1	0	0	1
0	1	1	0	×	×	×	×	×	0	1	0	1	0
0	1	1	1	0	×	×	×	×	0	1	0	1	1
0	1	1	1	1	0	×	×	×	0	1	1	0	0
0	1	1	1	1	1	0	×	×	0	1	1	0	1
0	1	1	1	1	1	1	0	×	0	1	1	1	0
0	1	1	1	1	1	1	1	0	0	1	1	1	1

由 74LS148 真值表（表 4-6）可知，该编码器有 8 个信号输入端、3 个代码输出端、1 个输入使能端、1 个输出使能端和 1 个输出扩展端。各端口情况如下：

(1) 74LS148 输入端为低电平有效，输出端以反码的形式表示。

(2) EIN 是输入使能端。低电平有效，即当 EIN=1 时，无论输入端是否有效，输出均呈现高电平，编码器处于非工作状态；而当 EIN=0 时，处于工作状态。

(3) EON 是输出使能端。当 EIN=0 且输入端无有效信号时，EON=0。故 EON=0 实际上是指示编码器处于工作状态，但此时“无编码信号输入”。

(4) GSN 是输出扩展端，低电平有效。即当 GSN=0 时，指示出编码器处于工作状态，且“有编码信号输入”。

由表 4-6 可见，当处于工作状态时，输入端 7N 的优先级别最高，0N 的优先级别最低。当 7N=0 时，其他 7 个输入端的数据都不会影响输出数据 ($A_2A_1A_0$)=000 的编码输出；而当 0N=0，其他输入端都为 1 时，输出数据 ($A_2A_1A_0$) 才有确定的编码 111 输出。根据表 4-6 能容易地绘出 74LS148 输入输出间的逻辑函数表示以及对应的电路图。

4.3.3 二-十进制编码器

二-十进制编码器就是用二进制码表示十进制数的编码器。由于十进制数中有 10 个数码，所以必须用 4 位二进制码才能完成对 10 进制数的十个数码进行编码。通常用 4 位二进制码组成 8421 BCD 码来表示十进制数，则二-十进制编码器的真值表如表 4-7 所示。根据表 4-7，对应的输出逻辑函数表达式为

$$D = I_8 + I_9$$

$$C = I_4 + I_5 + I_6 + I_7$$

$$B = I_2 + I_3 + I_6 + I_7$$

$$A = I_1 + I_3 + I_5 + I_7 + I_9$$

表 4-7 二-十进制编码器的真值表

输入	D	C	B	A
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

再根据上述各表达式可直接画出二-十进制编码器逻辑图，如图 4-10 所示。

常用的二-十进制优先编码器专用集成器件有 10-4 线 8421 BCD 编码器 74LS147，它的逻辑符号图如图 4-11 所示，其逻辑功能所对应的真值表如表 4-8 所示。

真值表 4-8 表明，74LS147 的优先级别从 9N 至 1N

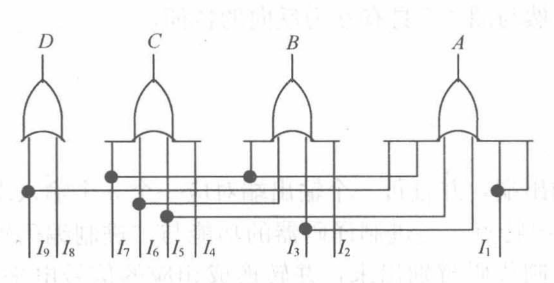


图 4-10 二-十进制编码器逻辑图

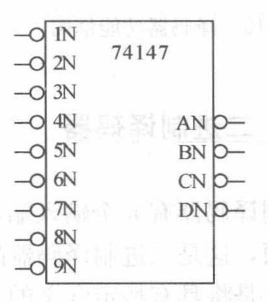


图 4-11 74LS147 的逻辑符号

递降，输入端为低电平有效，当输入端1N～9N均无效时，表示对0进行编码。输出端均以反码的形式出现。

读者不妨根据真值表4-8写出74LS147的逻辑函数，直至画出对应的逻辑电路。

表4-8　74LS147真值表

输入									输出			
9N	8N	7N	6N	5N	4N	3N	2N	1N	DN	CN	BN	AN
1	1	1	1	1	1	1	1	1	1	1	1	1
0	×	×	×	×	×	×	×	×	0	1	1	0
1	0	×	×	×	×	×	×	×	0	1	1	1
1	1	0	×	×	×	×	×	×	1	0	0	0
1	1	1	0	×	×	×	×	×	1	0	0	1
1	1	1	1	0	×	×	×	×	1	0	1	0
1	1	1	1	1	0	×	×	×	1	0	1	1
1	1	1	1	1	1	0	×	×	1	1	0	0
1	1	1	1	1	1	1	0	×	1	1	0	1
1	1	1	1	1	1	1	1	0	1	1	1	0

4.4　译　码　器

本节介绍译码器的基本概念，包括由门电路构成的二进制译码器及对应的集成标准译码器74LS138、二-十进制集成标准译码器74LS42和显示译码器集成电路。

4.4.1　译码器的概念

把具有特定含义的二进制代码“翻译”成数字或字符的过程称为译码，实现译码操作的电路称为译码器。译码器是一个多输入、多输出的组合逻辑电路，根据功能可分为二进制译码器、BCD码译码器和显示控制译码器等。译码器与编码器的功能恰好互为相反，因此它们对应的真值表中的输入输出变量和数据，在许多情况下，几乎能交换位置。图4-12为译码器功能框图，显然与图4-6具有互为反向的特征。

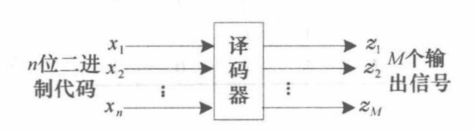


图4-12　译码器功能框图

4.4.2　二进制译码器

二进制译码器有n个输入端，2^n个输出端，并且每一个输出端对应一个n个输入端组成的最小项，这是二进制译码器的一个重要特点。二进制译码器的功能与二进制编码器刚好相反，它是将具有特定含义的不同二进制代码辨别出来，并转换成相应的信号电平。n个输入端组成的代码值确定后，2^n个输出端中总有一个（唯一的一个）为高电平（或低电

平），其余为低电平（或高电平）。常见的集成译码器有2-4线、3-8线和4-16线译码器等。

3位二进制（3-8线）译码器的真值表如表4-9所示，它有3个输入端A_2、A_1、A_0，用于输入3位二进制代码，有8个输出端$Y_0 \sim Y_7$，是8个互斥的信号。它是通过输出端的逻辑高电平来识别不同的输入代码的，这称为“输出高电平有效”。

由3-8线译码器的真值表4-9可得出输出逻辑函数的表达式：

$$Y_0 = \overline{A}_2\overline{A}_1\overline{A}_0 \qquad Y_1 = \overline{A}_2\overline{A}_1A_0 \qquad Y_2 = \overline{A}_2A_1\overline{A}_0$$

$$Y_3 = \overline{A}_2A_1A_0 \qquad Y_4 = A_2\overline{A}_1\overline{A}_0 \qquad Y_5 = A_2\overline{A}_1A_0$$

$$Y_6 = A_2A_1\overline{A}_0 \qquad Y_7 = A_2A_1A_0$$

以上的输出逻辑函数表达式表明，3位二进制译码器的输出端包含了3个输入端变量A_2、A_1、A_0组成的所有最小项。

3位二进制（3-8线）译码器74LS138的逻辑符号如图4-13所示。图中C、B、A为二进制译码输入端（它们分别对应表4-9中的A_2、A_1、A_0），Y0N～Y7N为译码输出端（输出低电平有效），G1、G2AN、G2BN为选通控制端。当G1＝1、G2AN＋G2BN＝0时，译码器处于工作状态；当G1＝0或G2AN＋G2BN＝1时，译码器处于禁止状态。读者不妨根据表4-9画出由门电路构成的3位二进制译码器的逻辑电路图，再针对74138的G1、G2AN、G2BN选通控制端的控制电平的要求列出74LS138的真值表，并画出74LS138内部的逻辑电路图。

表4-9 3位二进制译码器真值表

A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

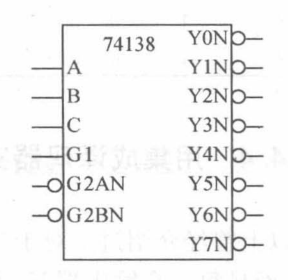


图4-13 74LS138逻辑符号图

4.4.3 二-十进制译码器

将输入的4位8421 BCD码翻译成0～9共10个十进制数的电路称为二-十进制译码器。二-十进制译码器有4个输入端和10个输出端。分别用A_3、A_2、A_1、A_0和$Y_0 \sim Y_9$表示。由于二-十进制译码器有4个输入端，10个输出端，所以又称为4-10线译码器。常用的集成二-十进制（4-10线）译码器是74LS42，其逻辑图如图4-14所示。

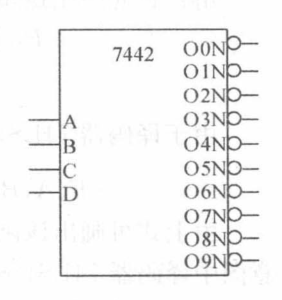


图4-14 74LS42逻辑图

表4-10是4-10线译码器74LS42的真值表。输入是8421BCD代码D、C、B、A，分别对应A_3、A_2、A_1、A_0；输出O0N～

O9N 分别对应 $Y_0 \sim Y_9$。输出的 $Y_0 \sim Y_9$ 为“输出低电平有效”，与表 4-9 的输出表达方式相反。

当 74LS42 输入无用码（伪码）1010～1111 时，输出 $Y_0 \sim Y_9$ 都为高电平 1，不会出现低电平 0。因此，译码器不会产生错误译码。由表 4-10 可以得到 74LS42 的输出逻辑表达式：

$$Y_0 = \overline{\overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}};\quad Y_1 = \overline{\overline{A_3}\,\overline{A_2}\,\overline{A_1}A_0};\quad Y_2 = \overline{\overline{A_3}\,\overline{A_2}A_1\overline{A_0}}$$

$$Y_3 = \overline{\overline{A_3}\,\overline{A_2}A_1A_0};\quad Y_4 = \overline{\overline{A_3}A_2\overline{A_1}\,\overline{A_0}}$$

$$Y_5 = \overline{\overline{A_3}A_2\overline{A_1}A_0};\quad Y_6 = \overline{\overline{A_3}A_2A_1\overline{A_0}};\quad Y_7 = \overline{\overline{A_3}A_2A_1A_0}$$

$$Y_8 = \overline{A_3\overline{A_2}\,\overline{A_1}\,\overline{A_0}};\quad Y_9 = \overline{A_3\overline{A_2}\,\overline{A_1}A_0}$$

表 4-10　4-10 线译码器 74LS42 的真值表

十进制数	A_3	A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

4.4.4　用集成译码器实现逻辑函数

以上曾经介绍过，对于二进制译码器，其输出为输入端的全部最小项（或最小项的非），而且每一个输出端 Y_i 为一个最小项（或最小项的非）。因为任何一个逻辑函数都可变换为最小项之和的标准式。因此，利用二进制译码器再加上必要的门电路，就可用于实现单输出或多输出的任何逻辑函数。

【例 4-4】试用译码器 74LS138 和与非门实现逻辑函数：$F(A, B, C)=AB+BC$。

解：首先写出逻辑函数 F 的最小项之和形式：

$$\begin{aligned} F(A,B,C) &= AB + BC = AB(C+\overline{C}) + (A+\overline{A})BC \\ &= AB\overline{C} + ABC + \overline{A}BC = \sum m(3,6,7) \end{aligned}$$

由于译码器 74LS138 的各输出端为最小项的非，故将上式再转化为如下形式：

$$F(A,B,C) = m_3 + m_6 + m_7 = \overline{\overline{m_3} \cdot \overline{m_6} \cdot \overline{m_7}} = \overline{Y_3 \cdot Y_6 \cdot Y_7}$$

由上式可画出该函数的电路图，如图 4-15（a）所示。其中 G1 接 1，两 G2 接 0。注意图中译码器 74LS138 本身的代码输入端 C、B、A 中 C 为最高位（根据 74LS138 的真值表），但该函数的输入变量 A、B、C 中 A 为最高位。

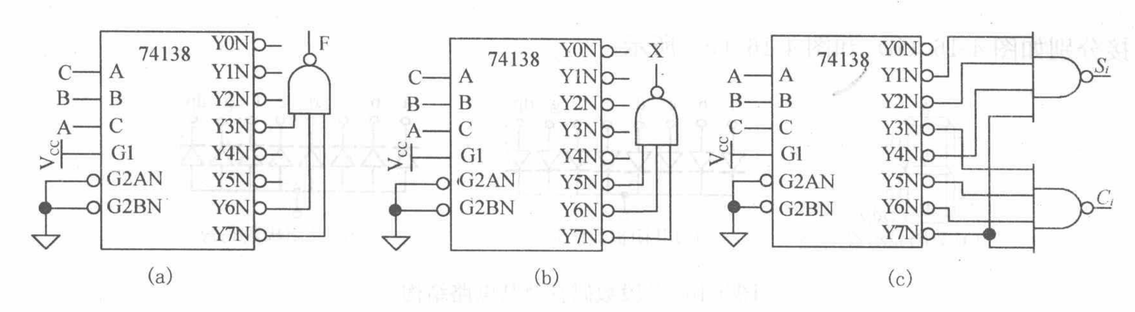


图 4-15 由 74138 构建的逻辑电路

用74LS138和与非门同样能实现例4-3设计的逻辑。根据真值表4-3得到输出逻辑函数，用最小项表示成以下形式：

$$X(A,B,C)=A(\overline{B}C)+A(B\overline{C})+A(BC)=m_5+m_6+m_7=\overline{\overline{m_5}\cdot\overline{m_6}\cdot\overline{m_7}}=\overline{Y_5\cdot Y_6\cdot Y_7}$$

按照以上逻辑式，将3-8译码器的输入接成 $C=A$、$B=B$、$A=C$，并将输出的 $\overline{m_5}$、$\overline{m_6}$、$\overline{m_7}$ 接与非门的输入端，即可实现所需逻辑函数的功能，电路连接如图4-15（b）所示。

显然，例4-3可以有多种实现方案，答案不唯一，实现方案应根据需要决定。

【例4-5】 有真值表如表4-11所示，用74LS138和与非门设计对应的逻辑电路。

解：写出输出最小项表达式，再转换成与非形式：

$$S_i=\overline{C}\,\overline{B}A+\overline{C}B\overline{A}+C\overline{B}\,\overline{A}+CBA=m_1+m_2+m_4+m_7$$

$$=\overline{\overline{m_1}\cdot\overline{m_2}\cdot\overline{m_4}\cdot\overline{m_7}}=\overline{Y_1\cdot Y_2\cdot Y_4\cdot Y_7}$$

$$C_i=\overline{C}BA+C\overline{B}A+CB\overline{A}+ABC=m_3+m_5+m_6+m_7$$

$$=\overline{\overline{m_3}\cdot\overline{m_5}\cdot\overline{m_6}\cdot\overline{m_7}}=\overline{Y_3\cdot Y_5\cdot Y_6\cdot Y_7}$$

由上述两式可画出逻辑电路图如图4-15（c）所示。

表 4-11 例 4-5 真值表

C	*B*	*A*	*S*	*C*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4.4.5 显示控制译码器

显示控制译码器可用于驱动数码显示器，这是一种可将二进制代码表示的数字、文字或符号用人们习惯的形式直观地显示出来的电路。

1. 7段数码显示器

7段数码显示器能够显示十进制或十六进制数字及某些简单字符。

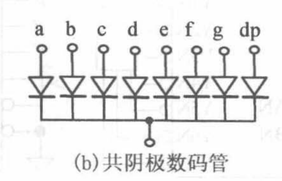
常见的7段数码显示器有发光二极管（Light Emitting Display，LED）和液晶显示器（Liquid Crystal Display，LCD）两种类型。这种数码管的每个线段都是一个发光二极管（或液晶显示段），因此也称LED数码管或LED七段显示器。一个7段数码管内部是由8个发光二极管组成的，其中7个发光二极管构成字型“8”的各个笔划（a～g），另一个发光二极管作为小数点（dp），外部结构和各对应段如图4-16（a）所示。

数码显示器根据公共端的连接方式，可以分为共阴极数码管和共阳极数码管，电路连

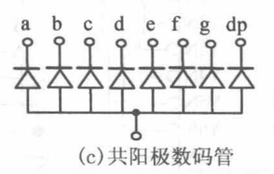
接分别如图 4-16（b）和图 4-16（c）所示。



(a)数码显示器



(b)共阴极数码管



(c)共阳极数码管

图 4-16　7 段数码管及其电路结构

以共阴极数码管为例，其公共端接低电平 0，当在某一段发光二极管（如 d 段）上施加正向电压（1）时，该段二极管即被点亮；不加电压则为暗。若要显示带小数点的 5，即“5.”，则在 dp、g、f、e、d、c、b、a 各段加上正向电压电平的情况是：“11101101”。为了保护各段 LED 不因电流过大而损坏，需在各个段上外加限流电阻保护，阻值大约为 100～510Ω。共阳极 7 段 LED 显示 0～F 的编码表如表 4-12 所示（以 dp 为最高位，a 为最低位）。对应的十进制数 0～9 的显示效果如图 4-17 所示。

表 4-12　共阳极数码管段选码表

显示字符	dp	g	f	e	d	c	b	a	段选码
0	1	1	0	0	0	0	0	0	C0H
1	1	1	1	1	1	0	0	1	F9H
2	1	0	1	0	0	1	0	0	A4H
3	1	0	1	1	0	0	0	0	B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H
6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	F8H
8	1	0	0	0	0	0	0	0	80H
9	1	0	0	1	0	0	0	0	90H
A	1	0	0	0	1	0	0	0	88H
B	1	0	0	0	0	0	1	1	83H
C	1	1	0	0	0	1	1	0	C6H
D	1	0	1	0	0	0	0	1	A1H
E	1	0	0	0	0	1	1	0	86H
F	1	0	0	0	1	1	1	0	8EH

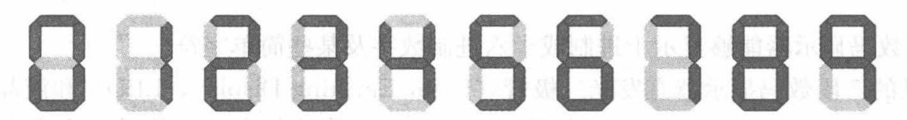


图 4-17　十进制数的显示效果

2. 7 段显示译码器

对应于不同极性的数码管，7 段显示译码器也有两种：第一种是输出为低电平有效，

可用于共阳极数码管的显示，如74LS47；第二种是输出为高电平有效，可与共阴极数码管搭配使用，如74LS48、CD4511（CMOS器件）等。

74LS48的逻辑符号如图4-18所示。其真值表如表4-13所示。由表4-13可知：

图4-18 74LS48的逻辑符号

（1）显示测试输入LTN。当LTN=0时，数码管的7段全亮，而与输入信号无关。此输入端用于测试数码管的好坏。

（2）动态灭0输入RBIN。低电平有效。当输入全为0时，如果LTN=1，RBIN=0，此时输出不显示，即“0”字被熄灭；如果LTN=1，RBIN=1，则输出正常显示“0”。而当输入不全为0时，输出正常显示。该输入端常用于消隐无效的“0”。如数据“00304.50”可显示为304.5。

表4-13 74LS48真值表

十进制数	输入						BI/RBO	输出						
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0
1	1	×	0	0	0	1	1	0	1	1	0	0	0	0
2	1	×	0	0	1	0	1	1	1	0	1	1	0	1
3	1	×	0	0	1	1	1	1	1	1	1	0	0	1
4	1	×	0	1	0	0	1	0	1	1	0	0	1	1
5	1	×	0	1	0	1	1	1	0	1	1	0	1	1
6	1	×	0	1	1	0	1	0	0	1	1	1	1	1
7	1	×	0	1	1	1	1	1	1	1	0	0	0	0
8	1	×	1	0	0	0	1	1	1	1	1	1	1	1
9	1	×	1	0	0	1	1	1	1	1	1	0	1	1
消隐	×	×	×	×	×	×	0	0	0	0	0	0	0	0
脉冲消隐	1	0	0	0	0	0	0	0	0	0	0	0	0	0
灯测试	0	×	×	×	×	×	1	1	1	1	1	1	1	1

（3）灭灯输入和动态灭0输出BIN/RBON。这是一个特殊的控制端，有时用作输入或输出。当作为输入时，且BIN/RBON=0时，不管输入为何，数码管7段全灭；当作为输出时，受控于LTN和RBIN。

（4）正常译码显示。当LTN=1，BIN/RBON=1，RBIN=1（即3个控制端均无效）时，对输入为十进制数0～9的BCD码进行正常译码显示。

3. 多数码管动态扫描显示控制方法

每一个译码器控制一个数码管的电路结构属于静态显示控制方式。其优点是编程容易，控制简单，管理方便，但是却要占用较多的硬件电路资源和器件的端口资源。所以在用数码管显示的位数较多情况下，一般采用动态扫描显示方式。

在多位七段LED数码管显示中，为了简化电路，降低成本，通常将所有数码管的段

选线并联在一起，刚好由译码器的 8 个输出口来控制 8 个段。而公共端（共阳极或共阴极）则分别由相应的位选择信号来控制，以实现各个位的分时选通控制。

利用人的视觉延迟的特点，采用扫描的方式驱动多位 7 段数码管，可以节省驱动电路，降低功耗。在保证一定的扫描频率条件下，可得到较高的显示质量。各位 7 段 LED 数码管公用一个 BCD 7 段译码器 74LS48，每位 7 段 LED 数码管的公共端连接到位选择器，位选择器由 3-8 译码器 74LS138 提供选择信号，控制各位数码管的点亮。

【例 4-6】 用一片 74LS48 和 74LS138 实现 8 位数码管的 8 位十进制数显示。

解：图 4-19 给出了此例要求的电路。8 个共阴 7 段数码管的数据端分别并接于 74LS48 的译码输出端 OA～OG，显示数据从 74LS48 的数据输入端 D、C、B、A 输入，经过译码后同时送到所有数码管的字形段 a～g 上。8 个数码管的阴极分别由 74LS138 的 8 个输出端来控制，使它们依次轮流显示。为提高亮度，可在这 8 个通道中加入电流驱动器件，如 9012 三极管；74LS48 的输出端也可插入功率驱动器件，如 74LS245。

事实上在实际应用中，十六进制数的显示更为常用。即要求将输入为十六进制数 0～F，在 7 段数码管上显示出 0～F 的字形。但目前的 74 系列的集成电路中尚没有这样的器件，因此这必须用第 6 章介绍的基于计算机的自动设计方法来表达和实现。

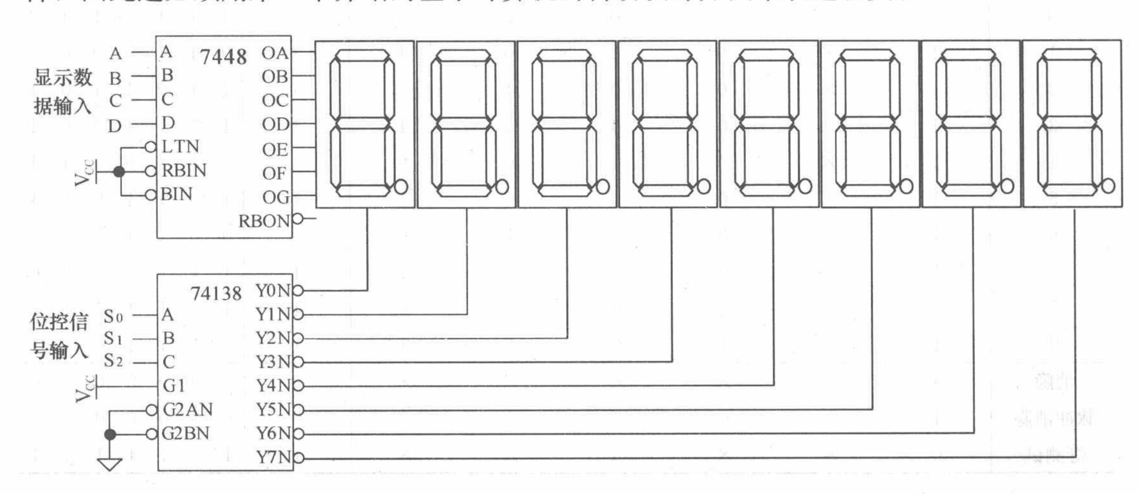


图 4-19　用 74LS48 和 74LS138 实现 8 位十进制数动态扫描显示

4.5　数据选择器与数据分配器

这里首先介绍数据选择器的基本概念及其应用和功能扩展的方法，然后介绍用数据选择器构成单输出的组合逻辑电路的设计方法，最后介绍数据分配器的功能。

4.5.1　数据选择器

数据选择器（Multiplexer）又称多路选择器或多路开关。其基本功能是，在地址信号（通道选择信号）控制下，从多路输入信号中选择其中一路数据送到输出口；这相当于一

个多输入的单刀多掷开关，结构如图 4-20 所示。2^n 个数据输入端的数据选择器必有 n 位地址输入端，称为 2^n 选 1 数据选择器。图 4-21 是 4 选 1 数据选择器。其中 D_0、D_1、D_2、D_3 为数据输入端，A_1、A_0 为地址输入端。地址变量 A_1、A_0 的取值决定从 4 路输入信号中选择输出其中的某一路。

根据 4 选 1 数据选择器的真值表（表 4-14）可获得其输出逻辑表达式如下：

$$Y = D_0\,\overline{A_1}\,\overline{A_0} + D_1\,\overline{A_1}A_0 + D_2A_1\overline{A_0} + D_3A_1A_0 = \sum_{i=0}^{3} D_i m_i$$

其中，m_i 是地址变量 A_1、A_0 组成的最小项，称为“地址变量最小项”。

由真值表可以看出，它实现了从多路输入端中选择其中某一个输入端数据作为输出的功能。根据真值表可以绘出对应的逻辑电路图。

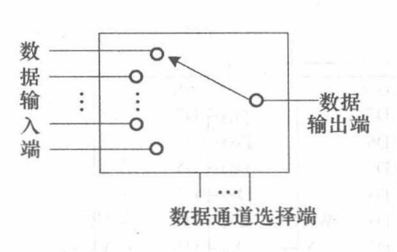


图 4-20 2^n 选 1 数据选择器示意图

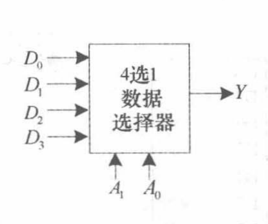


图 4-21 4 选 1 数据选择器逻辑符号

表 4-14 4 选 1 数据选择器真值表

输入			输出
A_1	A_0	GN	Y
×	×	1	0
0	0	0	D_0
0	1	0	D_1
1	0	0	D_2
1	1	0	D_3

传统设计中类似的常用标准集成数据选择器有：含 4 个 2 选 1 数据选择器的 74LS157；含 2 个 4 选 1 数据选择器的 74LS153；单 8 选 1 数据选择器 74LS151；16 选 1 数据选择器 74LS150 等。

这里以 74LS151 为例，讨论其功能和用法。74LS151 的逻辑符号如图 4-22 所示，其真值表如表 4-15 所示。

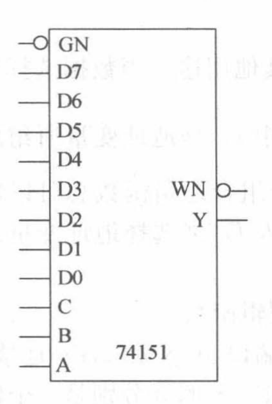


图 4-22 74LS151 逻辑符号

表 4-15 74LS151 的逻辑功能真值表

输入				输出	
A_2	A_1	A_0	GN	Y	W
×	×	×	1	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	1	0	D_1	$\overline{D_1}$
0	1	0	0	D_2	$\overline{D_2}$
0	1	1	0	D_3	$\overline{D_3}$
1	0	0	0	D_4	$\overline{D_4}$
1	0	1	0	D_5	$\overline{D_5}$
1	1	0	0	D_6	$\overline{D_6}$
1	1	1	0	D_7	$\overline{D_7}$

由真值表可获得 74LS151 的逻辑功能如下：

(1) GN=1 时，数据选择器被禁止，无论地址变量取何值，输出 Y 总是等于 0。

(2) GN=0 时，有

$$Y = D_0\overline{A}_2\overline{A}_1\overline{A}_0 + D_1\overline{A}_2\overline{A}_1A_0 + \cdots + D_7A_2A_1A_0 = \sum_{i=0}^{7}D_im_i$$

其中，m_i 是地址变量 A_2（即 C）、A_1（即 B）、A_0（即 A）组成的最小项。因此，输出 Y 提供了地址变量的全部最小项，这是数据选择器的一个重要特点。

为了实现多位数据的选择，可将几个一位数据选择器并联在一起，组成多位数据选择器。即将它们的使能端连在一起，相应的地址输入端连在一起。图 4-23 是将两个 8 选 1 数据选择器接成一个 2 位 8 选 1 数据选择器的电路。此外，还可进行数据选择器的扩展，如将两个 4 选 1 数据选择器连接成一个 8 选 1 数据选择器等。图 4-24 就是将两个 8 选 1 数据选择器连接成一个 16 选 1 数据选择器的电路。

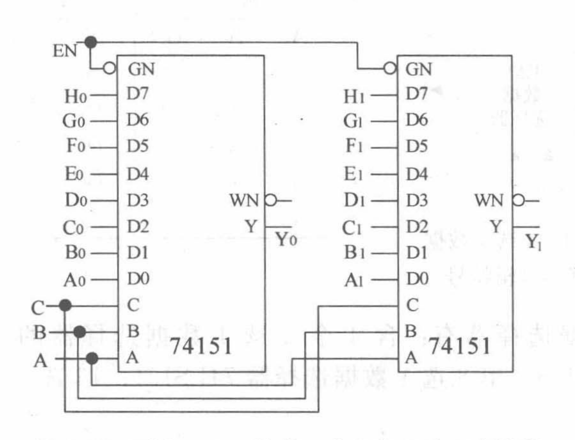


图 4-23　两个 74151 接成一个 2 位 8 选 1 选择器

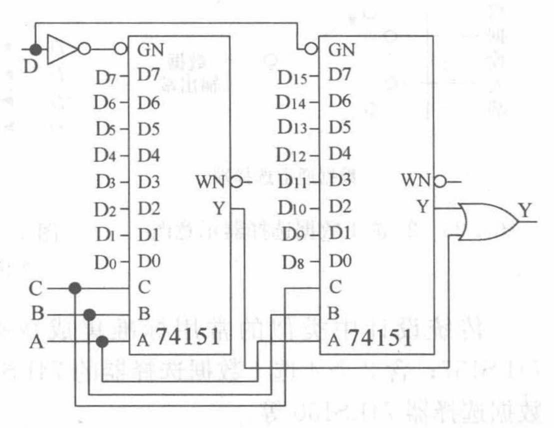


图 4-24　两个 74151 接成 16 选 1 选择器

4.5.2　用数据选择器实现逻辑函数

与译码器相同，数据选择器除本身的功能外，还可用于其他用途。当数据选择器的使能端有效时，它的输出逻辑函数表达式为 $Y = \sum_{i=0}^{2^n-1}D_im_i$ 。其中 m_i 是地址变量所组成的最小项（注意它与译码器输出端表达式的不同）。因为任何一个组合逻辑函数总可以用若干个最小项之和的标准形式构成。所以，利用数据选择器的输入 D_i 来选择地址变量组成的最小项 m_i，就可以实现任何组合逻辑函数。

【例 4-7】用 4 选 1 数据选择器 74153 模块实现例 4-3 的逻辑函数。

解：若采用 74153 内的一个 4 选 1 数据选择器，其输入端口 1GN 和 2GN 是禁止端，高电平时禁止工作，低电平允许工作；A、B 是通道选择端，1C0～1C3 分别是 4 个数据输入端；1Y 是对应的数据输出端。例如当 A=0，B=1，且正常工作状态下时，1C2 口的数

据通过 1Y 输出。于是 74153 中的 4 选 1 数据选择器 1 的功能表达式可写成

$$1Y = 1C0(\overline{B}\,\overline{A}) + 1C1(\overline{B}A) + 1C2(B\overline{A}) + 1C3(BA)$$

在上式中，如果把 B、A 作为两个输入的逻辑变量，同时在 1C0～1C3 接另一个逻辑变量的原变量或反变量，以及 0 或 1，就可以得到任何形式的三变量的逻辑函数。因此，可以将此 4 选 1 数据选择器作为一个三变量逻辑函数发生器来使用。根据真值表 4-3 对应的逻辑函数，则对应的逻辑形式可表为

$$X = 0(\overline{B}\,\overline{C}) + A(\overline{B}C) + A(B\overline{C}) + A(BC)$$

对比表 4-14 及其逻辑表述，可将 4 选 1 数据选择器的输入端按照如图 4-25 的方式连接，即：B=B、A=C、1C0=0、1C1=A、1C2=A、1C3=A，则获得表 4-3 的 Y 的逻辑函数（这里是 X）。

【例 4-8】 用 8 选 1 数据选择器 74LS151 实现逻辑函数：

$$L = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB$$

解：首先将组合逻辑函数变换成最小项之和的标准形式：

$$L = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

而 8 选 1 数据选择器输出信号的表达式为

$$Y = m_0D0 + m_1D1 + m_2D2 + m_3D3 + m_4D4 + m_5D5 + m_6D6 + m_7D7$$

比较 L 和 Y 可知：

$$D0 = 0 \quad D1 = 1 \quad D2 = 1 \quad D3 = 0$$
$$D4 = 0 \quad D5 = 0 \quad D6 = 1 \quad D7 = 1$$

由此可画出如图 4-26 所示的逻辑电路图。

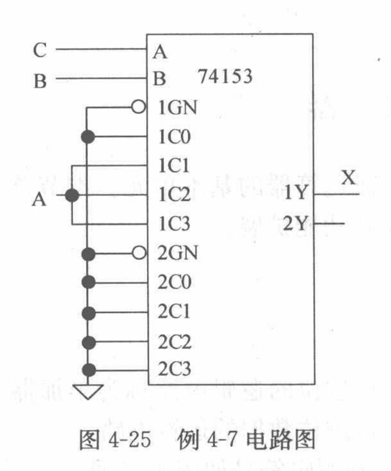


图 4-25 例 4-7 电路图

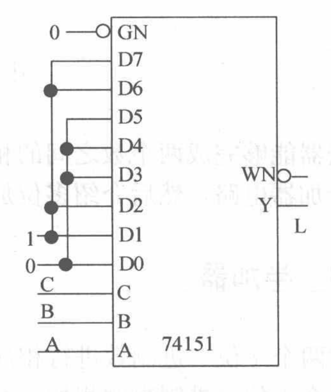


图 4-26 例 4-8 的电路图

注意： L 函数中最小项的最高位为 A，Y 函数中最小项的最高位为 C，两者要对应。

4.5.3 数据分配器

数据分配器能把一路输入端信号根据需要分配到多路输出中的某一路输出上。它的作用相当于一个多输出的单刀多掷开关。其结构示意图如图 4-27 所示。图 4-27 显示，通常的数据分配器只有一个数据输入端，有多个数据输出端。1-4 路数据分配器的逻辑符号如图 4-28 所示，其对应的真值表如表 4-16 所示。

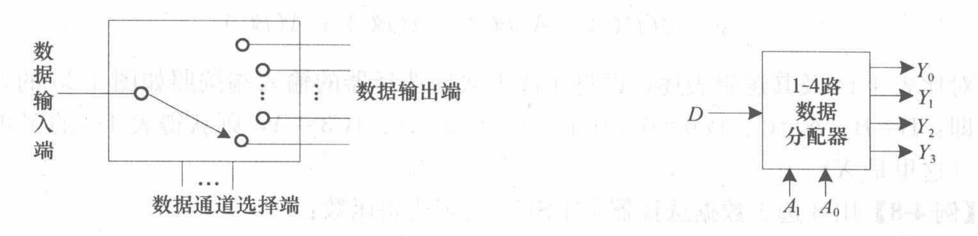


图 4-27　数据分配器示意图　　　　图 4-28　1-4 路数据分配器的逻辑符号

数据分配器可由带有使能输入端的二进制译码器来实现。如将译码器 74LS138 的使能端 G2BN 用作数据输入端，G1 接高电平，G2AN 接低电平，二进制代码输入端 C、B、A 作地址输入端时，则 74LS138 便构成了 1-8 路数据分配器，电路如图 4-29 所示。

表 4-16　1-4 路数据分配器真值表

输入		输出			
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	1	1	1	D
0	1	1	1	D	1
1	0	1	D	1	1
1	1	D	1	1	1

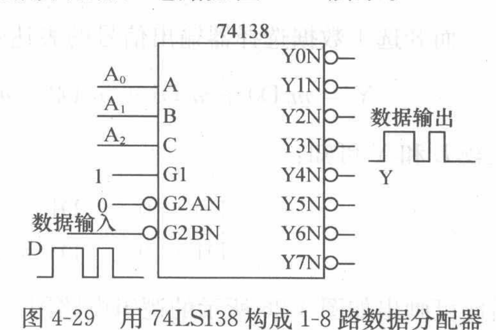


图 4-29　用 74LS138 构成 1-8 路数据分配器

4.6　加　法　器

加法器能够完成两个数之间的相加运算，是构成运算器的基本单元。本节首先介绍半加器及全加器电路，然后介绍多位加法器的应用及其功能扩展。

4.6.1 半加器

能对两个 1 位二进制数进行相加求和，并向高位进位的逻辑电路称为半加器。半加器只考虑两个 1 位二进制数的相加，而不考虑来自低位进位数据的运算电路。

半加器真值表如表 4-17 所示，对应的逻辑电路和逻辑符号如图 4-30 所示，其中 A、B 是加数，SO 是本位相加和值，CO 是向高位的进位数。

由真值表或逻辑电路图得到半加器的输出逻辑函数表达式为

$$S=\overline{A}B+A\overline{B}=A\oplus B \qquad C=AB$$

表 4-17 半加器真值表

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

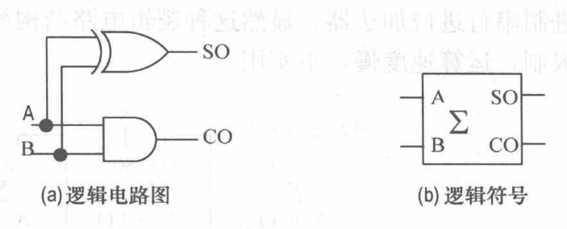


图 4-30 半加器的逻辑电路图和逻辑符号

4.6.2 全加器

能对两个 1 位二进制数进行相加并考虑来自低位的进位，求得和及向高位进位的逻辑电路称为全加器。全加器不仅考虑两个 1 位二进制数的相加，而且还把来自低位进位数也纳入相加的运算。全加器真值表如表 4-18 所示，其中 A_i、B_i 是加数，C_{i-1} 是低位的进位数，S_i 是本位和值，C_i 是向高位的进位数。由真值表 4-18 得到全加器的输出逻辑函数表达式：

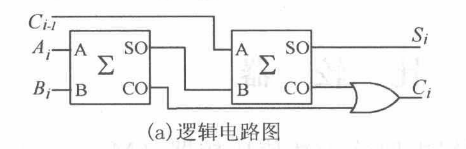
$$S_i=A_i\oplus B_i\oplus C_{i-1}$$

$$C_i=(A_i\oplus B_i)C_{i-1}+A_iB_i$$

图 4-31 所示的是全加器的逻辑电路图和逻辑符号。由图可见，全加器可以由两个半加器和一个或门组成。

表 4-18 全加器真值表

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



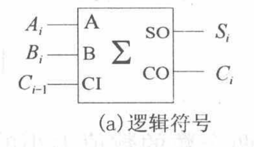


图 4-31 全加器的逻辑电路图和逻辑符号

4.6.3 多位加法器

实现多位二进制数相加的电路称为加法器。加法器按进位方式不同，可分为串行进位加法器和超前进位加法器。

1. 串行进位加法器

串行进位加法器是把 n 位全加器串联起来，低位全加器的进位输出连接到相邻的高位

全加器的进位输入。低位进位输出信号送给高位作为输入信号。因此任一高位的加法运算必须在低一位的运算完成之后才能进行，这种方式称为串行进位。图 4-32 所示的是 4 位二进制串行进位加法器。显然这种逻辑电路结构简单，但由于加法器的速度受到进位信号的限制，运算速度慢，不实用。

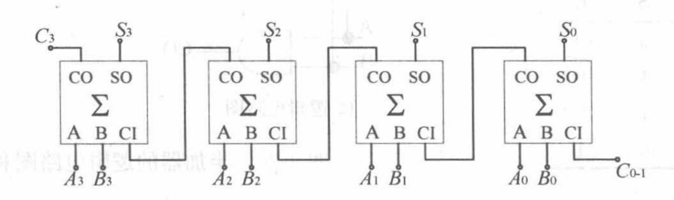


图 4-32 4 位二进制串行进位加法器

2. 并行进位加法器

由于串行进位加法器的运算速度慢，实际多采用并行进位加法器（超前进位加法器）。如在传统的手工设计技术中，常使用集成二进制 4 位并行进位加法器。常用的型号有 74LS83、74LS283 等。

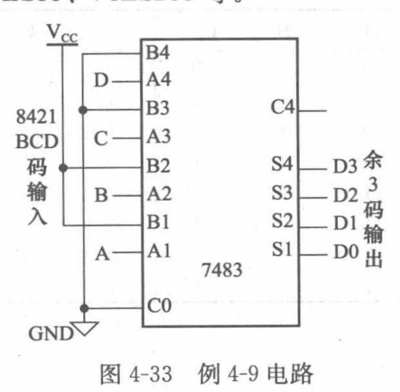


图 4-33 例 4-9 电路

加法器在逻辑电路中通常被用作运算电路，但也可作为逻辑代码转换电路。

【例 4-9】 试采用 4 位全加器 74LS83 完成 8421 BCD 码到余 3 码的转换。

解：由于 8421 BCD 码加 3（0011）即为余 3 码，所以其转换电路可以用加法器来实现。电路结构如图 4-33 所示。电路中 4 位 BCD 码 D、C、B、A 从端口（A4～A1）进入，二进制数 0011 从端口（B4～B1）进入。

4.7 比 较 器

用来比较两个数的数值大小的逻辑电路称为数值比较器（Magnitude Comparator）。比较结果有“大于”、“等于”和“小于”3 种情况。

4.7.1 1 位数值比较器

1 位数值比较器是比较器的基础，它只能比较两个 1 位二进制数的大小。它的逻辑图如图 4-34 所示。1 位数值比较器的真值表如表 4-19 所示，由此可得到它的输出逻辑表达式为

$$L_1 = A\overline{B} \qquad L_2 = \overline{A}B \qquad L_3 = \overline{A}\,\overline{B} + AB = \overline{\overline{A}B + A\overline{B}}$$

由真值表可知，将逻辑变量 A、B 的取值当作二进制数，当 $A>B$ 时 $L_1=1$；当 $A<B$ 时 $L_2=1$；当 $A=B$ 时 $L_3=1$。

表 4-19 1 位数值比较器的真值表

A	B	L_1 ($A>B$)	L_2 ($A<B$)	L_3 ($A=B$)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

图 4-34 1 位比较器逻辑图

4.7.2 集成数字比较器

多位数值比较器的设计原则是先从高位比起，高位不等时，数值的大小由高位确定。若高位相等，则再比较低位数。比较结果由低位的比较结果决定。常用的集成数值比较器有 74LS85、74LS518、74LS684 等。4 位比较器 74LS85 的逻辑图如图 4-35 所示。

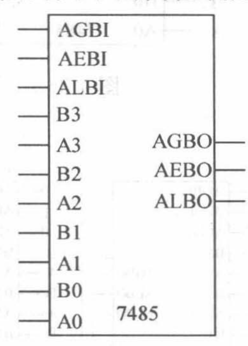


图 4-35 74LS85 的逻辑图

74LS85 的真值表如表 4-20 所示。真值表中的输入变量包括 8 个比较输入端 A_3、B_3、A_2、B_2、A_1、B_1、A_0、B_0 和 3 个级联输入端 AGBI ($A'>B'$)、ALBI ($A'<B'$) 和 AEBI ($A'=B'$)。级联输入端是为了便于输入低位数比较结果，并能与其他数值比较器连接，以便组成更多位的数值比较器。

3 个输出信号 AGBO ($A>B$)、ALBO ($A<B$) 和 AEBO ($A=B$) 分别表示本级的比较结果。

表 4-20 74LS85 功能真值表

比较输入				级联输入			输出		
A_3 B_3	A_2 B_2	A_1 B_1	A_0 B_0	$A'>B'$	$A'<B'$	$A'=B'$	$A>B$	$A<B$	$A=B$
$A_3>B_3$	×	×	×	×	×	×	1	0	0
$A_3<B_3$	×	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	0	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1	0	0	1

4.7.3 集成数值比较器应用举例

如果二进制数的位数比较多，就需将几片数值比较器连接进行扩展，数值比较器的扩展方式有并联和串联两种。图 4-36 为 2 片 4 位二进制数值比较器串联扩展为 8 位数值比较

器。图 4-37 是在 Quartus II 环境下调用的参数可设置宏模块（第 6 章及此后章节中将陆续介绍此类模块的使用）所设计的 8 位数值比较器。图 4-38 则为用 5 片 7485 并联扩展为 16 位数值比较器的电路。

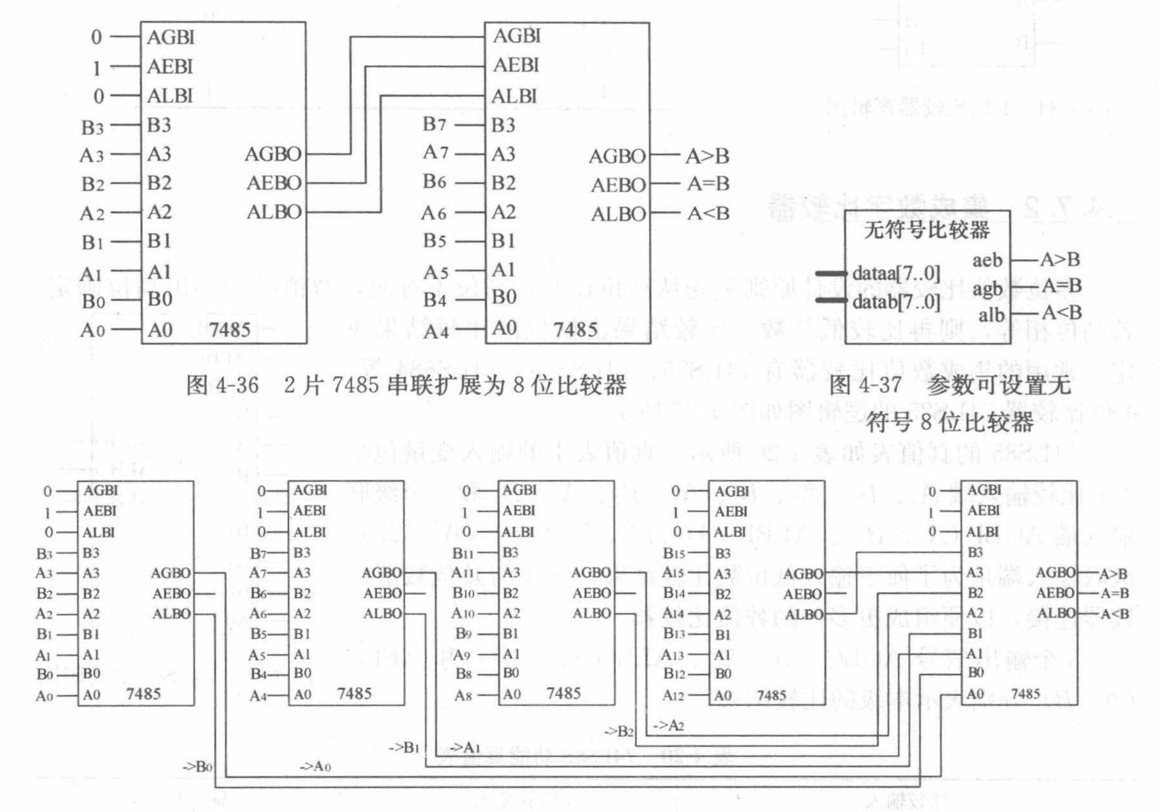


图 4-36 2 片 7485 串联扩展为 8 位比较器

图 4-37 参数可设置无符号 8 位比较器

图 4-38 5 片 4 位二进制数值比较器并联扩展成的 16 位比较器

4.8 广义译码器概念

尽管组合逻辑电路模块结构各异，功能繁多，但却有一个共同点，就是电路的输出数据总是当前输入数据的函数。或者说，在任何时刻电路的输出状态仅取决于此刻的输入状态，而与电路之前的状态无关。正是由于这种简单关系，我们可以暂时抛开它们特定的功能和名称，如“加法器”、“比较器”、“多路选择器”等，而仅用一张真值表来表达任何功能类型的组合电路，于是，数字电路设计者只需依据一张真值表即可对任何类型的组合电路模块进行分析和设计了。

可以这样来考虑，如果一个电路有 3 个数据输入端，2 个数据输出端，它们的组合逻辑关系可以表述为表 4-3 的形式（尽管它是一个表决器真值表）。在此典型的真值表中，左面列出输入信号的所有取值，右面列出输出信号的取值。于是类似的真值表所表达的可以是一个全加器，也可以是一个比较器，或者是一个特殊的译码器。任何功能的改变仅取决于输入或输出变量的数量和表中的数值。因此，真值表是任何组合电路设计所必须的，

且最基本的建模形式，是一切组合逻辑功能模块的抽象物。

从另一方面看，如果将真值表的输出数据看成是输入码（输入数据）所对应的编码，那么，所有组合电路模块的功能都能看成是一种译码行为。其输入的所有数据，无论是加数、被加数、通道选择控制信号、比较器数据还是使能控制数据，全部可以看成是一组待译码的编码，即输入数据，而输出的数据就是对应的译码数据。由此，我们就能将所有组合电路模块的功能都看成是一种译码行为。对于这种一般意义上的译码电路，我们权且称其为广义译码器。

这样一来，对于任何类型的组合电路的设计就归结为一个指定功能的广义译码器的设计，而一个针对广义译码器设计建模的关键是给出对应的真值表。

有了这张真值表，如果使用传统的手工设计实现方法，就可以沿用 4.2 节给出的流程来完成全部电路的设计与实现；如果使用现代自动设计技术，那么主要的手工工作就止于广义译码器真值表的抽象，或者说是编制。也就是说，对于自动设计流程，真值表一旦确定，余下的设计、分析和电路实现工作都可由计算机来完成了，于是电路性能和设计效率大为提高。

广义译码器的引入有利于在认识上将各类组合逻辑电路的设计简化成一张真值表的表达，同时使传统的数字技术概念和设计方法顺利地、平稳地、几乎无缝衔接地过渡到对现代自动设计技术的理解和把握，甚至包括对以后将要介绍的时序电路的结构、功能和设计的深入理解和高效设计奠定了可靠的基础。后面我们将会发现，在更一般的同步时序逻辑中，即有限状态机的结构中，这个广义译码器其实就是一个状态译码器。由此证明，无论是在组合电路还是时序电路，广义译码器具有意义深远的一般性含义、结构和功能。相关的内容将从第 6 章及后续章节展开。

4.9 可编程逻辑器件的结构与原理

在前面的章节中已经介绍了 74 系列等各种小规模数字逻辑集成电路，如译码器、编码器、加法器、比较器等逻辑器件。这种器件的逻辑功能都是固定不变的。

在利用固定功能逻辑器件进行数字电路设计时，通常将系统划分成若干个模块，然后选择合适的标准逻辑器件连接成所需的电路。然而这种设计方法存在许多缺陷，首先是电路的体积和功耗都可能很大，而且涉及的器件数量多，器件间的连线多，从而导致电路的速度低，可靠性差；其次是设计方案不便于修改，一旦形成电路，更改或修改的工作量会很大；而且设计人员需要非常熟悉大量的不同标准逻辑器件的性能和引脚封装，因此设计效率极低；设计完成后又很难对系统进行仿真、测试和修改。

此外，所能构建的逻辑规模小，实现的逻辑功能有限，特别是设计的电路容易被复制，不利于知识产权的保护。

所幸的是，可编程逻辑器件的出现彻底改变了这种不利的局面，并为电子设计自动化极大地扩充了可施展的领域。可编程逻辑器件（Programmable Logic Device，PLD）是早在 20 世纪 70 年代就发展起来的一种新的集成电路元件，是大规模集成电路技术发展的产物，是一种半定制的集成电路，即设计者可以根据需要对此器件的功能作进一步的设计和

重构。PLD集成了大量的逻辑单元和可编程连接资源，设计者可以利用计算机软件工具快速、方便地构建所需要的数字系统。设计者通过对器件中的逻辑单元的编程（功能重构），就可以方便地设计出具有各种不同逻辑功能的元件。

采用PLD设计逻辑电路可以充分发挥集成电路的一系列优势，利用电子系统自动设计软件能方便、快捷地完成对PLD的设计、编程、仿真和调试等工作，为设计工作带来极大的便利，同时最大程度地消除了基于传统的手工设计方法和标准逻辑器件构成的数字电路系统的种种不利因素。

为了能于第6章中尽快进入现代数字系统设计技术相关内容的介绍，本节的重点是在此前已给出的门电路知识的基础上，介绍可编程逻辑器件的结构和工作原理，初步展示数字系统自动设计技术重要的硬件平台。

4.9.1 PLD概述

曾经有两个重要的问题越来越严重地阻碍了数字电子设计技术的发展。其一是，随着电子技术的发展和市场对新的电子产品的要求，实用电子系统，特别是数字电子系统的规模越来越大，即使很一般的数字系统也已远超数千门的逻辑规模了。如果仍然使用诸如74系列的小规模逻辑器件来构建系统，势必导致产品的体积、成本和功耗急速上升，且由于一个产品中此类器件的数量太多而使系统的可靠性大为下降。其二则是，如果使用此类小规模逻辑器件，传统的数字电路手工设计技术显然无法完成涉及数千门以上的大规模逻辑电路的构建。其中一个明显的问题就是，74系列器件构成的逻辑系统，其硬件调试和测试便是一个不可逾越的障碍。因为如果在测试中发现速度、信号干扰、可靠性、甚至电路板布线方案等不符合设计要求时，只能放弃原来已完成的系统，从头开始设计。这将严重降低产品的设计效率，增加产品的设计成本，从而极大地损害产品的市场竞争力。

为了解决这些问题，人们提出了一种可编程逻辑器件的概念。这种器件是一种逻辑功能可通过计算机轻易重构的数字电路器件。然而，这种逻辑可再编程（重构）的器件，由于受到当时集成电路工艺技术的限制，一直未能如愿。直到20世纪后期，集成电路技术有了飞速的发展，可编程逻辑器件才得以实现。这种器件不仅能满足大规模，乃至超大规模（近千万逻辑门）的逻辑设计，更重要的是，最后获得的设计系统能通过计算机完成快速测试和电路结构重构（即电路的布线布局都能通过计算机快速完成），从而极大提高了数字系统的设计效率和系统性能，降低了设计成本。受此推动，这种基于计算机的数字系统自动设计技术（电子设计自动化EDA，Electronic Design Automation）也得到了迅猛的发展。

可编程逻辑器件的结构原理其实并不复杂。不论是简单还是复杂的数字电路系统都是由基本门来构成的，如与门、或门、非门、传输门等。

如前所述，由基本门可构成两类数字电路，一类称为组合电路，另一类称为时序电路。人们发现，不是所有的基本门都是必须的。例如，可用与非门单一基本门就可以构成其他任何基本门。任何组合逻辑函数都可以化为与或表达式，即任何的组合电路，都可以用与门-或门二级电路实现。同样，任何时序电路都可由组合电路加上存储元件，即锁存

器、触发器等构成。由此，可以提出一种可编程电路结构，即所谓“乘积项”逻辑可编程结构，其原理结构图如图 4-39 所示。

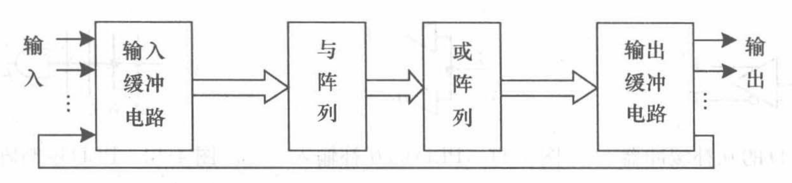


图 4-39 基本 PLD 器件的原理结构图

当然，“与-或”结构组成的 PLD 器件的功能比较简单。此后，人们又从数据只读存储器 ROM（Read Only Memory）工作原理、地址信号与输出数据间的关系，以及专用集成电路 ASIC（Application Specific Integrated Circuit）的门阵列法中获得启发，构造出另外一种可编程的逻辑结构，那就是所谓“查找表”的逻辑形成方法，它的逻辑函数的构成采用了静态随机存储器 SRAM“数据”查找的方式，并使用多个查找表构成了一个查找表阵列，称为可编程门阵列 PGA（Programmable Gate Array），从而设计出了更大逻辑规模和设计更灵活的 PLD 器件，即现场可编程门阵列 FPGA（Field Programmable Gate Array）。相关内容都将在以后的章节中介绍。

历史上，可编程逻辑器件经历了从可编程只读存储器 PROM（Programmable Read Only Memory）、可编程逻辑阵列 PLA（Programmable Logic Array）、可编程阵列逻辑器件 PAL（Programmable Array Logic）、通用可重复编程的阵列器件 GAL（Generic Array Logic），到采用大规模集成电路技术的可擦除型可编程逻辑器件 EPLD（Erazable Programmable Logic Device），直至复杂可编程逻辑器件 CPLD（Complex Programmable Logic Device）和 FPGA 的发展过程。在结构、工艺、集成度、功能、速度和灵活性方面都有了极大的改进和提高，并仍在不断地发展。

4.9.2 简单 PLD 的结构与工作原理

简单 PLD 是早期出现的可编程逻辑器件，它们的逻辑规模都比较小，只能实现小规模的数字逻辑电路设计（不大于 500 逻辑门），在结构上是由简单的与或门阵列和输入输出单元组成。常见的简单 PLD 有 PROM、PLA、PAL、GAL 等。

1. 电路符号表示

在介绍 PLD 原理之前，有必要熟悉一些描述 PLD 内部结构的常用电路符号。

由于 PLD 的特殊结构，用通常的逻辑门符号表示比较繁杂，特用一种约定的符号来简化表示。接入 PLD 内部的与或阵列输入缓冲器电路，一般采用互补电路结构表示，可用图 4-40 来表示，它等效于图 4-41 的逻辑结构，即当信号 A 输入 PLD 的一端后，分别以其同相信号 A 和反相信号 $\overline{A}$ 接入 PLD 器件中。

图 4-42 是 PLD 中与阵列的简化图形，表示可以选择 A、B、C 和 D 四个信号中的任一或全部输入与门。图中只标有 A、B、D 三根线被连向与门，因此逻辑输出是 $F=A\cdot B\cdot D$。

由此可见，在基本硬件配置不变的条件下，只须取舍连线间的连接节点就能轻易改变逻辑结构和逻辑函数，这就是最简单的可编程逻辑电路的实现原理。

图 4-40 PLD 的互补缓冲器　　图 4-41 PLD 的互补输入　　图 4-42 PLD 中与阵列的表示

由于图 4-42 中含有一个输入端可编程的与门，因此这个电路可以被看成一个可编程（电路结构可重构的）与阵列的最小单元。

注意在这里所表示的与阵列，是在原理上的等效。当采用某种硬件实现时，如用 NMOS 电路时，在图中的与门可能根本不存在，但 NMOS 构成的连接阵列中却含有了与的逻辑。同样，或阵列也用类似的方式表示，道理也相同。

图 4-43 是 PLD 中可编程或阵列的简化图形表示，根据连线的情况，可获得输出的逻辑函数是 $F=A+C$。图 4-44 是在阵列中连接关系的表示。十字交叉线表示此二线未连接；交叉线的交点上打黑点，表示是固定连接，即在 PLD 出厂时已连接；交叉线的交点上打叉，表示该点可编程（可重复断开和连上，且已连接上）。在 PLD 出厂后通过编程，其连接情况可根据设计者随时改变。图 4-42 中的 3 个连接点中，一个是固定连接（A 信号），两个（B、D 信号）是后来的编程连接。



图 4-43 PLD 中或阵列的表示　　图 4-44 阵列线连接表示

2. PROM

PROM 即可编程只读存储器，其除了用作只读存储器外，还可作为 PLD 使用。PROM 器件主要由地址译码部分、PROM 单元阵列和输出缓冲部分构成。为了更清晰直观地表示 PROM 中固定的与阵列和可编程的或阵列逻辑关系，可以用 PLD 阵列图来表达 PROM 的结构。如图 4-45 所示的是 PROM 中的部分结构。

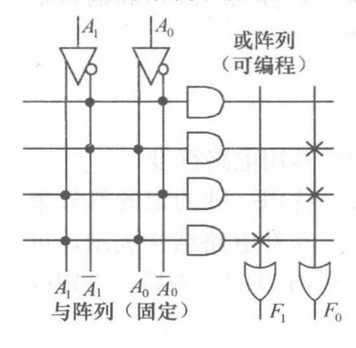


图 4-45 PROM 表达的 PLD 阵列图

有关 PROM 的更详细的内容将在第 10 章中介绍。

图 4-45 中，A_1 和 A_0 是两个输入 PROM 的信号，F_0 和 F_1 是输出信号。它们的逻辑关系可以通过观察其电路结构获得。图 4-45 中左侧是一个固定编程的与阵列，从所连的黑点可以知道进入 4 个与门的逻辑变量情况；而图右侧是可编程的或阵列，根据黑点和打叉点的情况，可以列出 F_0、F_1 与 A_1、A_0 的逻辑函数关系如下：

$$F_0 = A_0\overline{A_1} + \overline{A_0}A_1 \qquad F_1 = A_1A_0$$

即

$$F_0 = A_0 \oplus A_1 \qquad F_1 = A_0A_1$$

以上二组公式是图 4-45 结构的布尔函数表达式，左二

式即所谓的“乘积项”和的表达方式。由于可以将输入变量 A_1 和 A_0 的相与看成一个相乘的项，而把这些项的相或逻辑看成是加的关系，从而把这种逻辑表述称为乘积项表述。基于图 4-45 的可编程电路结构也就成了乘积项逻辑结构。右二式是对应的异或表达式。

不难发现，图 4-45 的电路结构形成了一个半加器的功能。由它们的逻辑表示可见，可以把式中的 A_1 和 A_0 分别看成是加数和被加数；F_0 和 F_1 分别看成是和值与进位。

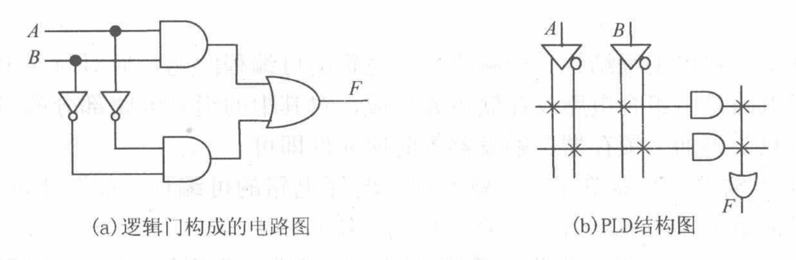
反之，如果要实现这个逻辑关系，或此半加器的功能，只须列出上述布尔函数，再由计算机根据 PROM 的结构，把此逻辑函数翻译成对应于此硬件结构的可编程阵列中连接点的烧写文件（阵列点文件）。于是通过针对 PROM 的烧写器，将此编程文件烧到 PROM 中，就能得到图 4-45 的连接结果，从而获得对应的逻辑关系。这就是利用可编程逻辑器件实现数字电路和数字系统自动化设计的基本思路。

这种由计算机根据设计者提供的逻辑函数生成的编程文件（阵列点文件），也称为熔丝图文件（Fuse Map）。目前对小规模可编程逻辑器件的设计和编程仍利用熔丝图文件来实现。显然，PROM 只能用于组合电路的可编程用途上，其输入变量的增加会引起存储容量的增加，这种增加是按 2 的幂次增加的，多输入变量的组合电路和时序逻辑都不适合用 PROM 来表达。

【例 4-10】 图 4-46（a）给出了函数 F 的逻辑图，假设与阵列和或阵列都是可编程的，试画出相应的 PLD 结构图。

解：根据给定的电路写出函数的逻辑表达式为 $F=AB+\overline{A}\,\overline{B}$。

按照 PLD 的表示方法，画出相应的 PLD 结构图如图 4-46（b）所示。



(a) 逻辑门构成的电路图　　(b) PLD结构图

图 4-46　函数 F 的逻辑结构图和 PLD 结构图

3. PLA

PROM 实现组合逻辑函数在输入变量增多时，其存储单元的利用效率将大大降低。PROM 的与阵列是全译码器，产生了全部最小项，在实际应用时，绝大多数组合逻辑函数并不需要所有的最小项。可编程逻辑阵列 PLA 对 PROM 进行了改进。由图 4-47 可见，PROM 的或阵列可编程，而与阵列不可编程。PLA 则是与阵列和或阵列都可编程，灵活性大增。图 4-47 是 PLA 的阵列图的部分电路结构表示。

4. PAL

PLA 的利用率很高，但是与阵列、或阵列都可编程的结构造成了电子设计自动化软件算法过于复杂，计算机运行速度下降。因此人们在.PLA 后又设计了另外一种可编程器件，即 PAL。PAL 的结构与 PLA 相似，也包含与阵列、或阵列。但是或阵列是固定的，

只有与阵列可编程（可重构的）。PAL 的基本结构如图 4-48 所示，由于 PAL 的或阵列是固定的，一般用图 4-49 来表示。

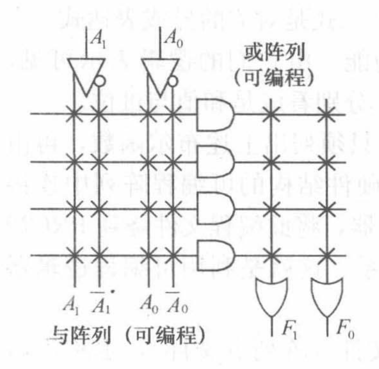


图 4-47 PLA 部分逻辑阵列示意图

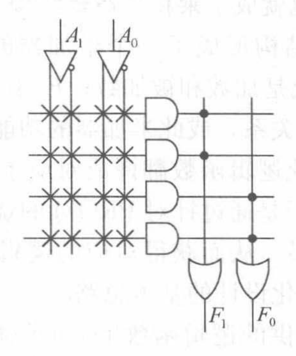


图 4-48 PAL 结构

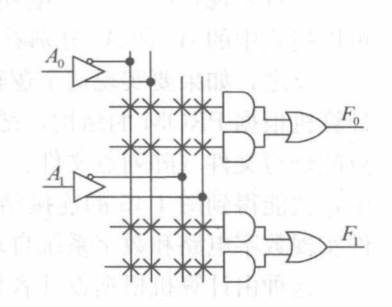


图 4-49 PAL 的常用表示

与阵列可编程，而或阵列固定的结构避免了 PLA 存在的一些问题，运行速度也有所提高，更适合于利用计算机来进行自动化设计。从 PAL 的结构可知，各个逻辑函数输出化简，不必考虑公共的乘积项。送到或门的乘积项数目是固定的，大大简化了设计的算法，同时也使单个输出的乘积项为有限。如图 4-49 中表示的 PAL 只允许有两个乘积项。对于多个乘积项，PAL 通过输出反馈和互连的方式解决，即允许输出端的信号再馈入下一个可编程与阵列。图 4-50 是型号为 PAL16V8 的 PAL 器件的部分结构图，从图中可以看到 PAL 的输出反馈。这种可编程结构的器件就比较适合于利用计算机来完成数字电路自动化设计。

上述提到的一些可编程结构只能解决组合逻辑的可编程问题，而对时序电路却无能为力。由于时序电路是由组合电路及存储单元构成，对其中的组合电路部分的可编程问题已经解决，所以只要再加上锁存器、触发器等时序元件即可。

PAL 加上了输出寄存器单元后，就实现了时序电路的可编程。如图 4-50 中的电路结构中就包含了 D 触发器（这将在第 5 章中详细介绍）。

这里将图 4-50 中的有关元件作一简要说明，这对进一步理解 PLD 可编程的原理是有帮助的。图左边有两个互补式输入口，引脚号分别是 2 和 3；器件结构内也有两个互补式接口元件，主要用于将输出口的信号反馈进可编程与阵列中。可编程与阵列共有 32 根纵线，因此中部的 16 个与门中的任一与门最多可以有 32 个逻辑变量输入。在每组 8 个与门输出后，进入一个 8 输入的或门（其中有一个与门的输出要通过一个多路选择器才进入或门），然后这个或门的输出与一个异或门的一端相连。此异或门的另一输入端由一信号 SL 控制。SL 能决定当或门输出的逻辑信号通过异或门后是同相输出还是反相输出。因为当设置 SL=0 时，将同相输出；当 SL=1 时将反相输出。而 SL 的电平由计算机根据逻辑设计程序来自动决定。显然，增加一个异或门能使基于计算机的逻辑反相要求的自动化设计变得更加容易实现。图 4-50 的右侧是两个 D 触发器，它是一个时序元件，另有 6 个多路选择器。它们的选择取向也由计算机来决定。在第 19、18 脚输出口处分别有一个三态控制门，可通过其控制端决定 19、18 脚端口是输出口还是输入口。显然，PAL 器件具有很大的逻辑表达灵活性和逻辑功能实现的通用性。

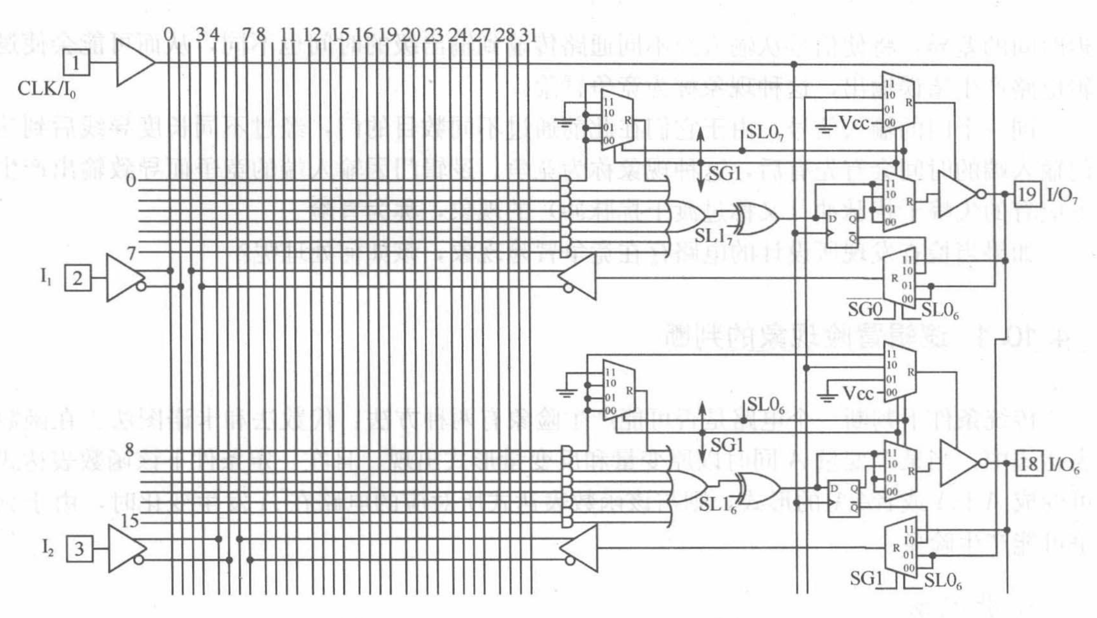


图 4-50 PAL16V8 内部的部分逻辑结构图

4.10 组合电路的竞争与冒险

信号在器件内部通过连线及逻辑单元时，都会有一定的延时。延时的大小与连线的长短、逻辑单元的数目、器件的制造工艺、工作电压和环境温度等因素都有关系，此外由于半导体器件内的载流子运行情况，信号的高低电平转换也需要一定的过渡时间。由于这些因素，多路信号的电平值发生变化的瞬间，组合逻辑的输出就有了先后顺序，这种非同时性变化，往往会出现一些不希望的尖峰信号，这些信号被称为“毛刺”或毛刺脉冲。

对于图 4-51 给出的电路，如果进入 A、B、C、D 4 个输入信号电平变换不是同时发生，这将导致输出信号 OUT 出现毛刺，仿真波形如图 4-52 所示。此外，由于无法保证所有内部连线的长度一致，即使 4 个输入信号在输入端同时变化，到达或门的时间也会不一样，从而也可导致毛刺的产生。如果将它们的输出直接连接到其他器件的控制输入端，有可能会导致错误后果。所以在数字电路设计完成后，必须检查设计系统的工作性能，其中包括检查输出信号是否含有毛刺。

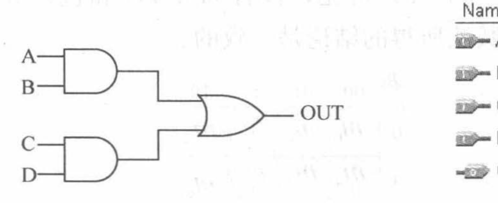


图 4-51 存在逻辑冒险的电路示例

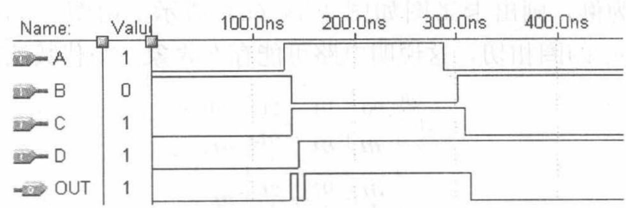


图 4-52 电路图 4-51 假设的仿真波形

此外由于从输入到输出的过程中，不同通路上的逻辑门数不同，或者是门电路平均延

迟时间的差异，将使信号从输入经不同通路传输到输出级的时间也不同，从而可能会使逻辑电路产生错误输出，这种现象称为竞争冒险。

同一个门的输入信号，由于它们在此前通过不同数目的门，经过不同长度导线后到达门输入端的时间会有先有后，这种现象称为竞争。逻辑门因输入端的竞争而导致输出产生不应有的尖峰干扰脉冲（又称过渡干扰脉冲）的现象，称为冒险。

如果当检查发现所设计的电路存在竞争冒险现象，该如何处理呢？

4.10.1 逻辑冒险现象的判断

传统条件下判断一个电路是否可能产生险象有两种方法：代数法和卡诺图法。在函数表达式中，当某个变量 A 同时以原变量和反变量形式出现，且在一定条件下该函数表达式可变成 $A+\overline{A}$ 或者$A\overline{A}$ 的形式，则与该函数表达式所对应的电路在 A 发生变化时，由于竞争可能产生险象。

1. 代数法

代数法是根据逻辑电路的结构来判断是否具有产生静态逻辑冒险的条件。具体方法是，首先检查某个输入变量是否同时以原变量和反变量出现在函数表达式中，如果该变量仅以一种形式出现，则它的变化不会引起静态逻辑冒险。若某变量有两种形式出现，则在不做任何化简的条件下，判断是否存在与其他变量的特殊组合，使函数变成 $A+\overline{A}$ 或者 $\overline{AA}$ 的形式，若存在这样的特殊组合，则说明电路可能会产生逻辑冒险。举例说明。

【例 4-11】 某逻辑函数表达式为 $F=A\overline{B}+BC$，试判断该逻辑电路是否可能产生冒险现象。

解：表达式中 B 以原变量和反变量的形式出现。假设输入变量 $A=C=1$，将 A、C 的值代入表达式，得 $F=\overline{B}+B$，理论上无论 B 为何值，该函数表达式 F 的值恒为 1。当 B 发生变化时，可能使电路产生冒险现象。

2. 卡诺图法

用卡诺图检查电路是否有可能产生险象比代数法更为直观和方便。具体方法是，首先画出函数的卡诺图，并画出函数表达式中各项所对应的圈。然后观察卡诺图是否存在某两个圈“相切”的情况。若存在相切的情况，则说明电路有可能产生逻辑冒险。仍以例 4-11 为例，画出卡诺图如图 4-53（a）所示。由图 4-53（a）可见，包含 m_4、m_5 和包含 m_3、m_7 的圈相切，这说明电路可能存在险象。与代数法所得的结论是一致的。

A \ BC	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4 1	m_5 1	m_7 1	m_6

(a)

A \ BC	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4 1	m_5 1	m_7 1	m_6

(b)

图 4-53 例 4-11 的卡诺图

4.10.2 冒险现象解决方法

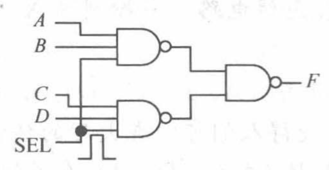
1. 增加冗余项

当卡诺图中有两个圈相切时，可能会产生冒险。如果在相切处增加一个圈，就可以消除冒险现象，所增加的乘积项称为冗余项。

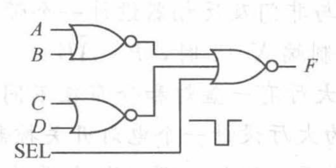
在例 4-11 中原函数表达式为 $F=A\overline{B}+BC$，如果将 m_5、m_7 圈起来，如图 4-53（b）中的虚线框所示，所对应的函数 $F=A\overline{B}+BC+AC$，比原来的函数多了一个冗余项 AC。当 $AC=11$ 时，$F=\overline{B}+B+1$ 输出保持为 1，从而消除了可能产生的险象。

2. 选通法

在电路中增加选通脉冲来避免冒险的发生，选通脉冲的极性和加入的位置应根据具体结构而定。在图 4-54（a）中，选通脉冲 SEL 采用高电平有效的形式，加在输入级与非门的输入端。当输入信号 A、B、C、D 变化时，选通信号 SEL=0，迫使电路的输出 $F=0$。当输入信号稳定以后，选通信号 SEL=1，电路输出正确的逻辑电平。



(a) 选通脉冲SEL采用高电平有效的电路



(b) 选通脉冲SEL采用低电平有效的电路

图 4-54　用选通脉冲避免冒险

在图 4-54（b）中，选通脉冲 SEL 采用低电平有效的形式，加在输出级或非门的输入端。当输入信号变化时，选通信号 SEL=1，迫使电路的输出 $F=0$。当输入信号稳定以后，选通信号 SEL=0，电路输出正确的逻辑电平。

冒险现象和毛刺电平问题的解决一直是数字逻辑设计工程中十分棘手却又必须面对的问题。以上给出的解决方法只能针对低速小规模逻辑电路的部分设计情况，至于现代高速大规模数字系统，其问题远非仅通过卡诺图或逻辑方程就能发现和解决的。更加深入的问题将于第 8 章中讨论。

习　题

4-1　分析图 4-55，写出 F 的表达式，并用最少的与非门实现其功能。

4-2　分析图 4-56 所示电路的功能。图中 S_1、S_0 为控制信号，A、B 为输入信号，Y 为输出信号。

4-3　已知逻辑电路如图 4-57 所示，分析该电路的逻辑功能。

4-4　试分析图 4-58 中用 74LS138 译码器构成的逻辑电路，写出输出端 F 的逻辑表达式，列出真值表，说明电路的逻辑功能。

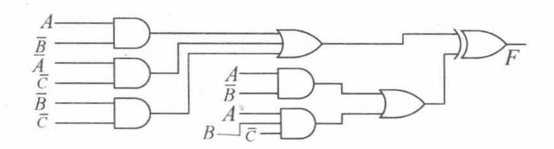


图 4-55 题 4-1 逻辑图

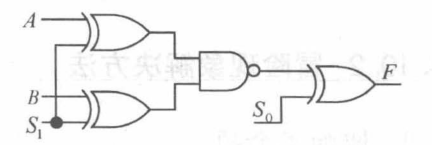


图 4-56 题 4-2 电路

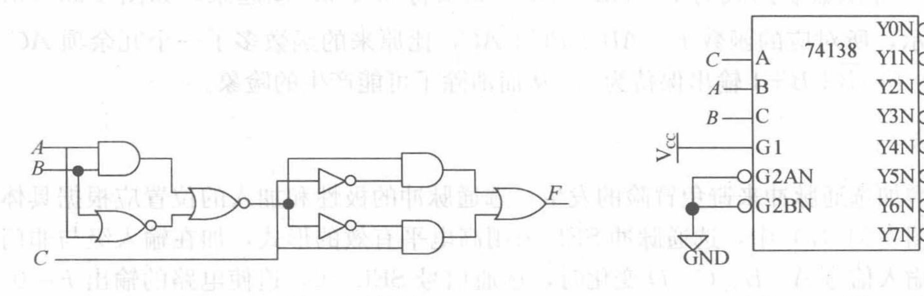


图 4-57 题 4-3 逻辑图

图 4-58 题 4-4 逻辑图

4-5 用与非门及反相器设计一个带控制端的组合逻辑电路，当控制端 $X=0$ 时，$F=A\oplus B$；当控制端 $X=1$ 时，$F=\overline{AB}$。

4-6 某大厅有一盏灯和分布在不同位置的 4 个开关（A、B、C、D）。试利用 4 选 1 数据选择器为大厅设计一个电灯开关控制逻辑电路，使得人们可以在大厅的任何一个位置控制灯的亮或灭。例如：可以用 A 开关打开，然后用 B（或 C、D、A）开关熄灭。

4-7 试用 4 选 1 数据选择器实现下列逻辑函数。

(1) $Y(A,B,C,D)=A\overline{B}D+\overline{A}B\overline{C}D+BC+\overline{B}C\overline{D}$

(2) $F(A,B,C)=\sum m(1,2,3,4)$

4-8 试用 8 选 1 数据选择器 74LS151 实现下列逻辑函数。

(1) $Y=A\overline{B}\,\overline{C}+A\overline{B}C+\overline{A}\,\overline{B}C$

(2) $Y=A\odot B\odot C$

(3) $Y=A\oplus B\oplus C$

(4) $F(A,B,C,D)=\sum m(1,5,6,7,9,11,12,13,14)$

(5) $F(A,B,C,D)=\sum m(0,2,3,5,6,7,8,9)+d(10,11,12,13,14,15)$

4-9 用 3-8 线译码器 74LS138 和与非门实现下列多输出函数：

$$F_1=AB+\overline{A}\,\overline{B}\,\overline{C} \qquad F_2=A+B+C \qquad F_3=\overline{A}B+A+\overline{B}$$

4-10 用 74LS148 和与非门实现 8421 BCD 优先编码器。

4-11 用 74LS139 组成一个 5-24 线译码器。

4-12 用二片 8-3 线优先编码器 74LS148 扩展为 16-4 线优先编码器，并分析其工作原理。

4-13 用 74283 加法器和逻辑门实现 1 位 8421 BCD 码加法器电路，输入输出均是 BCD 码，C_I 为低位的进位信号，C_O 为高位的进位信号，输入为两个 1 位十进制数 A，输出用 S 表示。

4-14 设计一个组合电路，用来判断输入的 4 位 8421 BCD 码 A、B、C、D，当其值大于或等于 5 时，输出为 1，反之输出为 0。

4-15 设计一个能实现两个 1 位二进制数全加器和全减器的组合逻辑电路。

4-16 用 7 个开关控制一个灯。当 k1、k3、k5 和 k7 闭合而 k2 断开；或者 k2、k4 和 k6 闭合而 k3 断开时，灯亮。试用或非门设计这个程控线路。

4-17 由双 4 选 1 数据选择器 74153 构成的电路如图 4-59 所示，其内部的单 4 选 1 数据选择器的真值表如表 4-21 所示。试写出 F 的表达式，并用最小项之和 $\sum m$ 的形式表示。

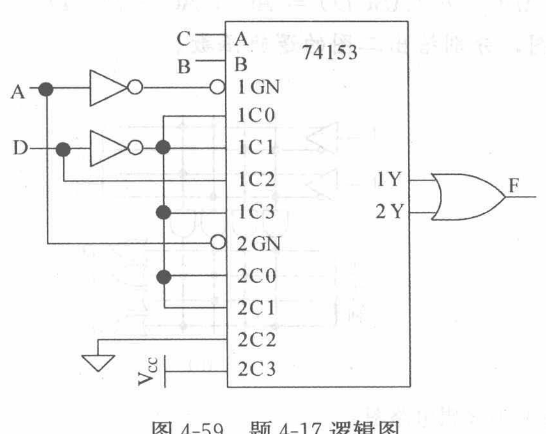


图 4-59 题 4-17 逻辑图

表 4-21 (1/2) 74153 功能表

GN	B	A	Y
1	x	x	0
0	0	0	C_0
0	0	1	C_1
0	1	0	C_2
0	1	1	C_3

4-18 由 74LS151 构成的电路如图 4-60 所示，试写出该电路输出函数 Y 的逻辑表达式，以最小项之和形式表示。若实现逻辑函数 $Y=\sum m(1,2,5,7,8,10,14,15)$，则电路作何改动？

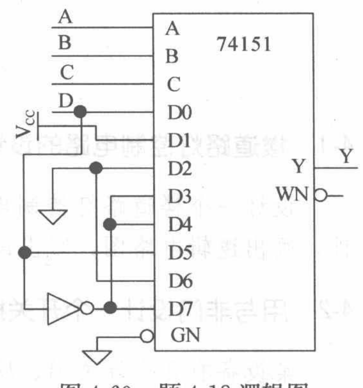


图 4-60 题 4-18 逻辑图

4-19 设计一个 4 位变补电路。提示：即输入一个 4 位二进制数，将此数据逐位取反再加 1，然后输出。

4-20 设计一个 4 位二进制数变余 3 码和一个余 3 码转换成 8421 BCD 码的转换电路。

4-21 用译码器 74138 和适当的逻辑门实现函数 $F=\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+AB\overline{C}+ABC$。

4-22 用 4 选 1 数据选择器实现组合逻辑：$L(A,B,C)=\overline{A}\,\overline{B}C+\overline{A}B\overline{C}+AB$。要求根据题意确定输入变量和输出变量。写出三输入逻辑变量的真值表，并分析输入输出逻辑关系。由真值表写出逻辑表达式 $L(A,B,C)$，并用卡诺图化简。使用双 4 选 1 数据选择器 74LS153，画出逻辑电路图。

4-23 仿照全加器的设计方法，设计一个半减器和全减器，所用的门电路自定。

4-24 用两片 3-8 译码器 74LS138 扩展为 4-16 译码器。要求给出逻辑电路图，硬件验证 4-16 译码器的逻辑功能，并说明电路的工作原理。

4-25 设计一个能比较一位二进制数 A 与 B 大小的比较电路。用 L_1、L_2、L_3 分别表

示3种状态，即 L_1（$A>B$），L_2（$A=B$），L_3（$A<B$）。写出设计全过程。

4-26 引入广义译码器的意义是什么？

4-27 什么是基于乘积项的可编程逻辑结构？

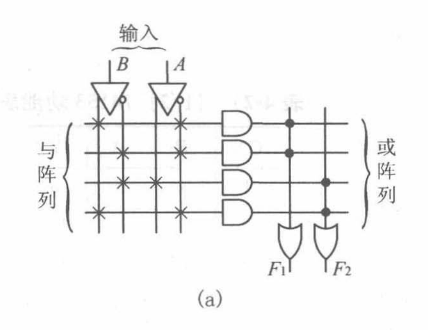
4-28 用PROM设计一个4位二进制码转换为格雷码的代码转换电路。二进制码转换为格雷码的真值表可参考表1-4。

4-29 试用PROM设计组合逻辑电路，画出相应的电路。已知函数 $F_1 \sim F_4$ 为

$$F_1(ABCD)=\overline{A}\,\overline{B}+\overline{B}\,\overline{D}+A\overline{C}D+BCD \quad F_2(ABCD)=\overline{AD}+B\,C\overline{D}+A\,\overline{BCD}$$

$$F_3(ABCD)=\overline{A}\,B\overline{C}+\overline{A}CD+A\overline{C}D+ABC \quad F_4(ABCD)=A\overline{C}+\overline{A}C+\overline{B}+\overline{D}$$

4-30 根据图4-61中的二个逻辑电路图，分别给出二图的逻辑函数。



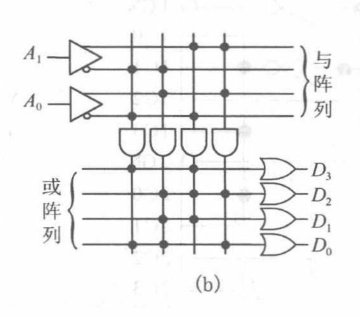


图4-61 题4-30逻辑电路图

实 验

4-1 楼道路灯控制电路的设计

设计一个楼道路灯控制电路，要求在3个不同的地方都能控制这盏灯。完成理论设计，画出逻辑电路图，写出设计全过程，在实验箱上进行验证，完成实验报告。

4-2 用与非门设计一个开关控制的报警电路

某设备有3个开关 A、B、C，当开关 A 接通时开关 B 才能接通，开关 B 接通时开关 C 才能接通，违反操作规程，则发出报警信号。要求写出设计的全过程，画出设计电路图并在实验箱上验证。

第5章 触发器及含触发器的PLD

数字系统中除了组合逻辑电路外，还需要有具备存储功能的电路。触发器就是实现存储功能的一种基本单元电路。触发器和组合逻辑电路相结合可以构成寄存器、计数器等时序逻辑电路。本章将详细介绍几种常用触发器的电路结构、逻辑功能、基本特点及其应用，以及触发器间的相互转换。最后介绍含有触发器，即时序元件的可编程逻辑器件PLD的结构原理，以便为进一步介绍面向时序逻辑电路设计的数字系统自动设计技术作必要的知识准备。

5.1 概 述

为了设计各种实用的数字系统，构建完备功能的逻辑电路，除了需要实现逻辑运算的组合逻辑门之外，还需要有能够保存信息的逻辑器件。触发器就是一种具有记忆功能的电子器件，是时序逻辑电路的基本元件，触发器的特点如下：

① 通常情况下，触发器有两个互补的输出端Q和$\overline{Q}$。

② 触发器有两个稳定状态。通常定义输出端Q=1、$\overline{Q}$=0时，称为1状态；而当Q=0、$\overline{Q}$=1时称为0状态。当输入信号不发生变化时，此状态稳定不变。

③ 在某些特定输入信号的作用下，触发器可以从一种稳定状态转移到另一种稳定状态。当此输入信号撤销后，将保持新的状态不变。通常把此输入信号作用之前的状态称为“现态”，记作Q^n和$\overline{Q}^n$；而把输入信号作用后的状态称为触发器的“次态”，记作Q^{n+1}和$\overline{Q}^{n+1}$。为简单计，一般省略现态的右上标n，就用Q和$\overline{Q}$表示现态。显然，次态是现态和输入信号的函数。

由上述特点说明，触发器是存储1位二进制信息的理想器件。

集成触发器的种类很多，分类方法也各不相同，若根据触发器的逻辑功能来分类，通常可分为RS触发器、D触发器、J-K触发器和T触发器4种类型。不论如何分类，就其结构而言，触发器都是由逻辑门加上适当的反馈通道耦合而成的。本节将从实际应用出发，介绍几种常用集成触发器的内部结构、工作特性和逻辑功能，重点讨论它们的逻辑功能及其描述方法。

含有触发器的逻辑电路称为时序逻辑电路，其特性结构决定了电路具有如下特征：

① 电路由组合电路和不同类型的触发器电路组成，具有对过去输入保持记忆的功能。而纯组合电路的输出仅取决于当前的输入信号。

② 如果考虑触发器内部电路的话，电路中一定包含反馈回路。

③ 电路的输出由电路当时的输入情况和状态，即现态共同决定。

5.2 RS触发器

在经典的数字电路中，尽管极少看到直接将RS触发器用作时序元件的情况，但它却是学习和认识触发器的入门器件。本节主要介绍RS触发器的电路结构和逻辑功能。

5.2.1 基本RS触发器

基本RS触发器又称置0、置1触发器。可以认为，它是构成各种功能触发器的最基本的单元，也称为基本触发器。RS触发器的基本特征是，内部电路通过交叉连接产生了正反馈。RS触发器有如下两种实现方法：

① 由两个或非门交叉连接而成的高电平输入有效型RS触发器，其逻辑电路如图5-1（a）所示，对应的逻辑符号如图5-1（b）所示。

② 由两个与非门交叉连接而成的低电平输入有效型RS触发器，其逻辑电路如图5-1（c）所示，对应的逻辑符号如图5-1（d）所示。

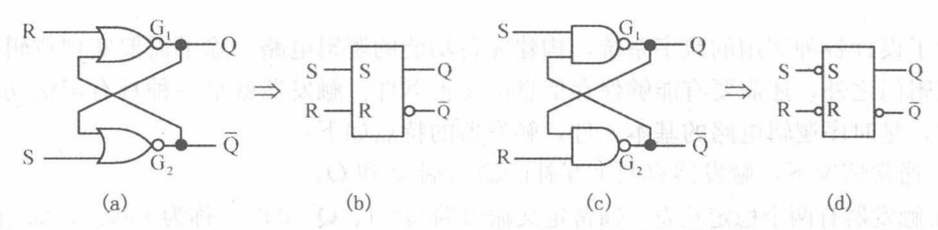


图5-1 用两种不同逻辑门组成的基本RS触发器及其逻辑符号

在图5-1中，Q和$\overline{Q}$是两个互补的输出端，并且定义Q=0，$\overline{Q}$=1为触发器的0状态，Q=1，$\overline{Q}$=0为触发器的1状态。以图5-1（a）所示的RS触发器为例，根据输入信号R、S的不同取值组合，触发器的输出与输入之间的关系有以下4种情况：

(1) 当S=0，R=0时，这两个输入信号对或非门的输出Q和$\overline{Q}$不起作用，触发器维持原状态不变，称为保持。这体现了触发器具有记忆功能，即在此条件下，触发器输出端的Q和$\overline{Q}$上各自“记住”了原有的状态值1或0。

(2) 当S=0，R=1时，无论原来Q、$\overline{Q}$处于何种状态，因R=1使得或非门G_1输出Q=0，则$\overline{Q}$=1，即触发器为0状态。这种情况称为触发器置0或称触发器复位，因此R输入端称为置0输入端或复位端。

(3) 当S=1，R=0时，无论原来Q、$\overline{Q}$处于何种状态，因S=1使得或非门G_2输出$\overline{Q}$=0，Q=1，这种情况称为触发器置1或置位，所以S输入端称为置1输入端或置位端。

(4) 当S=1，R=1时，Q=$\overline{Q}$=0，触发器的两输出互补的逻辑关系被破坏。而且当两个输入信号都同时撤去（注意，必须是同时撤去，即S和R同时变到0）后，触发器的状态将不能确定是1还是0，这是一种不稳定的状态。在实用中，这种情况应当避免。

综上所述，图5-1（a）所示的基本RS触发器的真值表应如表5-1所示。从真值表中

可看出，触发器的输入端为高电平有效。表5-1中，当S和R同时为1时，Q和$\overline{Q}$输出都为0* 的含义是，S和R同时脱离1后，触发器的状态不定，而非指此时的状态不定。

对于图5-1（c）中的RS触发器来说，这是由与非门构成的触发器，这种触发器是以低电平作为输入有效信号的，在逻辑符号的输入端用小圆圈表示低电平输入信号有效，它的真值表如表5-2所示。由表5-2可以看出，由于S=0，R=0时出现了Q=1*，$\overline{Q}$=1* 的状态，表示此时输出尽管为1，但当S和R同时撤去（变到1时）后，触发器的状态也将不能确定是1还是0，因此也应当避免出现这种情况。

表5-1　或非门组成的RS触发器的真值表

R	S	Q	$\overline{Q}$	触发器状态
0	0	不变	不变	保持
0	1	1	0	置1
1	0	0	1	置0
1	1	0*	0*	不定

表5-2　与非门组成的RS触发器的真值表

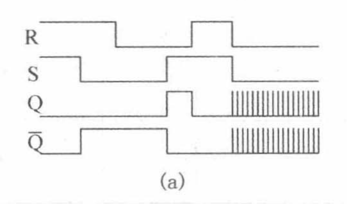
R	S	Q	$\overline{Q}$	触发器状态
0	0	1*	1*	不定
0	1	0	1	置0
1	0	1	0	置1
1	1	不变	不变	保持

图5-2是基本RS触发器的仿真波形，其中图（a）是或非门RS触发器的工作波形，图（b）是由与非门构成的RS触发器的工作波形。此波形图来自Quartus II软件。

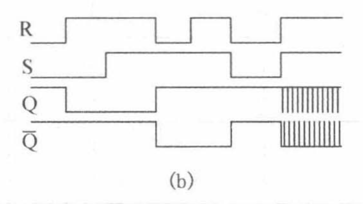
观察图5-2（a）可以发现，当R=1和S=1时，Q和$\overline{Q}$都为0；而在之后，当R和S同时为0时，输出的状态都呈现高速跳动的脉冲，表示无法确定它们此时的电平。这完全符合对应的图5-1的电路图及其真值表（表5-1）。

通过图5-2（a）这张仿真波形图，可以更好地理解真值表（表5-1）中的最后一行的含义。最后一行的含义是这样的，当R和S都为1时，Q和$\overline{Q}$肯定都为稳定的0（现态）；但当从当前情况下，R和S都为1时的情况同时撤去（即同时变到0）后，Q和$\overline{Q}$的状态（次态）便处于不确定的值。所以，表中的"不定"并非是指当R和S都为1时的当前的情况，而是之后的状态不定。对于图5-2（b）的波形分析也相同。

另外请特别注意，由于图5-2的波形来自对图5-1电路的时序仿真，即考虑了硬件延迟特性的仿真，输出相对于输入信号在时间上有明显的滞后。



(a)



(b)

图5-2　两种基本RS触发器的仿真波形图

5.2.2 具备时钟控制的RS触发器

为了克服基本RS触发器直接控制的缺点，在基本RS触发器基础上增加两个控制门和一个时钟脉冲信号，使触发器采用同步控制。同步的含义就是只有在时钟脉冲（简称为

时钟，用 CP 表示）信号到来时触发器的输出才会改变。或者说，触发器输出的改变与时钟是同步的。这种具备时钟信号控制电路的 RS 触发器称为钟控 RS 触发器。钟控触发器按逻辑功能来分类，可分为 RS、D、JK、T 等类型触发器。

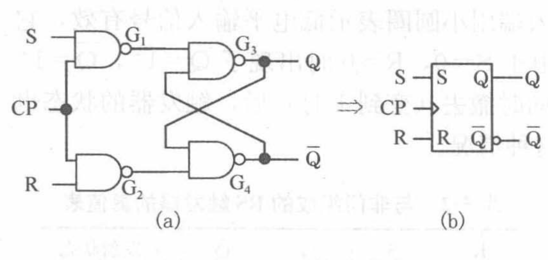


图 5-3 钟控 RS 触发器

钟控 RS 触发器由基本 RS 触发器构成，电路结构如图 5-3（a）所示，其对应的逻辑符号如图 5-3（b）所示。G_1、G_2 门组成控制门，G_3、G_4 门组成基本 RS 触发器。时钟信号通过控制门，控制输入信号 R、S 进入 G_3 和 G_4 门的输入端。电路功能分析如下：

当 CP=0 时，G_1、G_2 门禁止，输入信号 R、S 不会影响输出端的状态，故触发器保持原状态不变。

当CP=1 时，G_1、G_2 门启动，R、S 信号通过 G_1、G_2 门反相后加到由 G_3、G_4 门组成的基本 RS 触发器上，此时工作情况与基本 RS 触发器相同。

根据以上分析得到钟控 RS 触发器的真值表如表 5-3 所示。由于触发器在每次时钟脉冲触发后产生的新状态 Q^{n+1}（次态）不仅与输入信号有关，而且还与触发器在每次时钟脉冲触发前的状态 Q^n（现态）有关，所以在表 5-3 中列入了 Q^n 和 Q^{n+1}。因此，这里把这种含有 Q^n 和 Q^{n+1} 变量的真值表叫做触发器的状态转换真值表。

表 5-3 钟控 RS 触发器状态转换真值表

CP	S	R	Q^n	Q^{n+1}	功能说明
0 0	× ×	× ×	0 1	0 1	$Q^{n+1}=Q^n$ 保持
1 1	0 0	0 0	0 1	0 1	$Q^{n+1}=Q^n$ 保持
1 1	0 0	1 1	0 1	0 0	$Q^{n+1}=0$ 置 0
1 1	1 1	0 0	0 1	1 1	$Q^{n+1}=1$ 置 1
1 1	1 1	1 1	0 1	1* 1*	不允许

这种次态、现态与输入信号之间的逻辑关系也可用特性方程来描述。根据真值表（表 5-3），得到钟控 RS 触发器的特性方程如下：

$$\begin{cases} Q^{n+1} = S + \overline{R}Q^n \\ RS = 0 \quad \text{（约束条件）} \end{cases} \tag{5-1}$$

虽然钟控 RS 触发器没有单独的集成电路芯片，但它是构成 D 触发器、JK 触发器等常用触发器的基础。

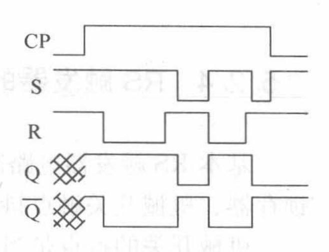


图 5-4 RS触发器空翻波形图

同步钟控 RS 触发器无疑克服了基本 RS 触发器的不足。它利用 CP 脉冲来选通控制，实现 CP＝0 时触发器被禁止，CP＝1 期间接收输入。

但是，在 CP＝1 期间仍是直接控制，如果此期间输入信号发生多次变化，触发器状态也将随之多次变化（如图 5-4 所示），这称为空翻。注意图 5-4 中打叉图像是输出电平不确定情况。为了克服空翻现象，必须使 RS 输入信号有特定的约束，即要求每来一个 CP 脉冲，触发器的状态只允许改变一次。解决空翻的方法就是采用以下要介绍的主从结构触发器和边沿触发器等。

5.2.3 主从RS触发器

图 5-5 给出的是由两个钟控 RS 触发器组成的主从 RS 触发器的电路。其工作原理简述如下。

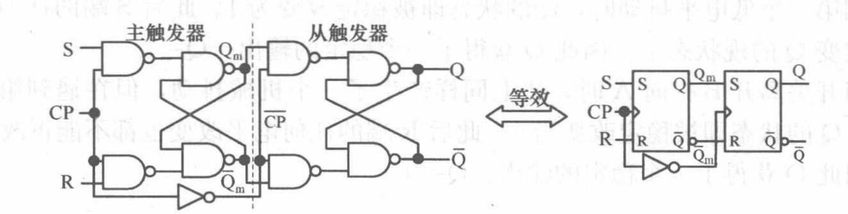


图 5-5 主从 RS 触发器

(1) CP＝1 期间：主触发器工作，其输出状态按照下面特性方程变化：

$$\begin{cases} Q_m^{n+1} = S + \overline{R}Q_m^n \\ RS = 0 \end{cases} \tag{5-2}$$

上式中，S、R 为 CP＝1 期间的输入信号，故主触发器还是存在“空翻”的可能。而从触发器保持输出状态不变。

(2) CP 由 1 变为 0，即下降沿到来时：主触发器保持 CP＝1 期间的最后输出状态不变，并作为从触发器的输入；同时，从触发器开始工作，由于主触发器的两个输出始终相反，故从触发器的输出状态跟随主触发器的最后输出状态（根据钟控 RS 触发器的真值表 5-3 得到）。故有

$$\begin{cases} Q^{n+1} = Q_m^{n+1} = S + \overline{R}Q_m^n = S + \overline{R}Q^n \\ RS = 0 \end{cases} \tag{5-3}$$

(3) CP＝0 期间：即使 S、R 输入信号发生变化，主触发器的状态继续不变，于是从触发器的输入不变，故从触发器保持上述动作后的输出状态不变，从触发器无空翻。

综上所述，在一个时钟周期内，主触发器可能发生多次翻转，但从触发器只发生一次翻转，故整个主从 RS 触发器克服了空翻现象。当然，其缺点仍然明显，即输入信号 R、S 仍然存在约束条件 RS＝0。

5.2.4 RS 触发器的应用

基本 RS 触发器电路简单，是构成各种性能完善的集成触发器的基础电路。如可构成锁存器、机械开关触点抖动消除电路或单脉冲发生电路等。

机械开关的特点是当开关从一个位置扳到另一个位置时，在固定接触之前会发生数次物理震动或抖动。虽然这些抖动间隔非常短暂，但它们可以产生瞬间电压峰值而形成“毛刺”脉冲，影响脉冲识别。

采用基本 RS 触发器构成的机械开关抖动消除电路如图 5-6（a）所示，结合图 5-6（b）对应时刻中的开关工作过程和 RS 触发器的逻辑功能，容易了解抖动消除电路的工作原理：

（1）由于 R、S 端都处于上拉状态，即在没有外来信号时都能处于稳定的高电平，即能保持原有的状态。假如这时处于低电平的开关处于 A 点，此时 R 输入低电平，因此 Q 稳定在 0。

（2）当开关离开 A 打向 B 时（接 S），S 上将产生一个机械抖动，由于 RS 触发器的特性，在遇到第一个低电平抖动时，Q 的状态即被稳定改变为 1。此后 S 端的任何电平改变都不能再改变 Q 的现状态了。因此 Q 获得了一个稳定的输出：Q=1。

（3）当开关离开 B 打向 A 时，R 上同样产生了一个机械抖动，但在遇到第一个低电平抖动时，Q 的状态即被稳定改变为 0。此后 R 端的任何电平改变也都不能再改变 Q 的现状态了。因此 Q 获得了一个稳定的输出：Q=0。

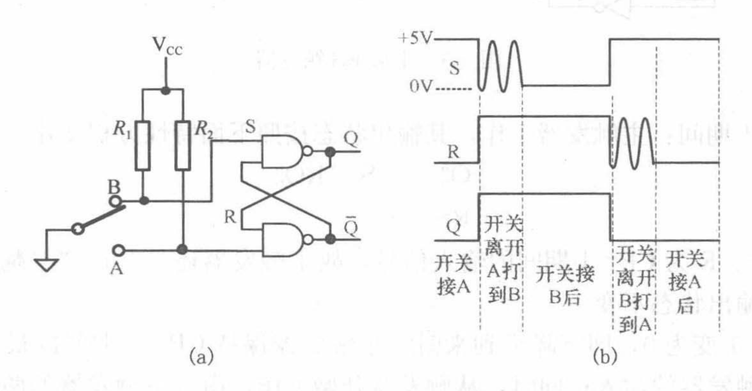


图 5-6　开关触点抖动消除电路

图 5-6 的电路不仅可以消除开关的抖动，而且从波形可以看出，此电路还可作为手动单次脉冲产生电路。

5.3　D 触发器

常用的触发器有多种类型，而在传统数字系统设计中，较多用到 JK 型触发器，因此对 JK 型触发器有更多的关注和介绍。但由于集成电路技术的发展和数字自动化设计技术的进步，端口比较简单的 D 型触发器的应用已处于绝多优势，甚至绝大多数国际流行的基

于 EDA 的集成电路设计软件时序元件库的基本触发器，或标准单元都采用 D 触发器。或者说，在基于自动化设计的数字系统中，几乎所有时序电路的底层时序元件都是 D 触发器。为此，本章将重点放在 D 触发器的结构原理及其应用的介绍上。

5.3.1 最简结构电平触发型 D 触发器

5.2 节的讨论表明，无论是基本 RS 触发器还是主从 RS 触发器，其共同的缺点是，存在状态不确定的可能，从而使其输入信号 R、S 存在约束条件，这在实用中是很不方便的。

图 5-7（a）给出了一种触发器的电路结构，十分简单，却解决了 RS 触发器的缺点。这个电路可以称为最简结构的电平触发型 D 触发器，现代数字设计中称其为 D 型锁存器（D Latch)，也有称其为透明型触发器的。

图 5-7（a）的 ENA 相当于 CP，在这里的作用则可看成是数据 D 输入的使能控制，当 ENA＝1 时，Q＝D，即数据输入端的 D 能通过电路向 Q 输出，这时 D 与 Q 是直通的，这就是所谓的“透明”；当 ENA＝0 时，电路保持原来的状态，即 Q 不随 D 的变化而变化。于是，分析电路图 5-7（a）可得到表 5-4 所示的电平触发型 D 触发器的真值表，由真值表即可得到对应的特性方程：

表 5-4 D 触发器真值表

D	Q^n	Q^{n+1}
0	0	0
1	0	1
0	1	0
1	1	1

$$Q^{n+1} = D \tag{5-4}$$

特性方程表明，触发器次态的值取决于现态时刻的 D 端的数据。

图 5-7（b）是此触发器的逻辑符号，图 5-7（c）的时序仿真波形反映了触发器的许多逻辑特性。图 5-7（c）显示，当 CLK＝1 时，Q 随着 D 的变化而变化；当 CLK＝0 时，Q 的值取决于 CLK 下降沿前一刻的 D 的电平。

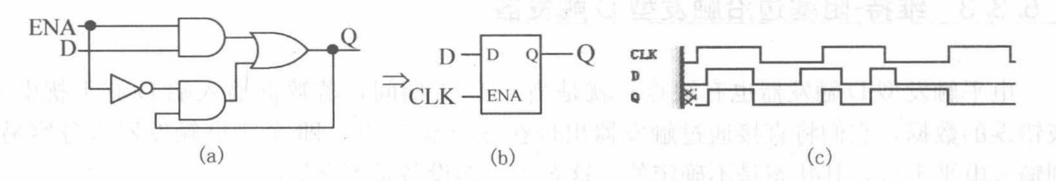


图 5-7 最简结构电平触发型 D 触发器及其时序仿真波形

5.3.2 经典结构电平触发型 D 触发器

这里介绍一种比较经典的 D 触发器电路，它包含了一个基本 RS 触发器，整个电路结构颇似钟控 RS 触发器。为了解决钟控 RS 触发器的 R、S 之间的约束问题，对钟控 RS 触发器稍做修改，即将其 R 端接至 G_1 门的输出端，并将 S 改为D，于是变成了图 5-8（a）所示的电路结构形式，这样便成为只有一个输入端的 D 触发器。其逻辑符号如图 5-8（b）所示。此 D 触发器在时钟脉冲作用期间（CP＝1 时），将输入信号转换成一对互补信号，送至基本 RS 触发器的两个输入端，使基本 RS 触发器的两个输入信号只能是 01 或者是 10

两种组合，从而消除了状态不确定的现象，解决了对输入的约束问题。

由于此D触发器是在CP=1时控制D触发器的状态变化，所以同样是电平触发型D触发器。当CP=0时，触发器被禁止，输入信号不起任何作用，其状态始终保持不变。

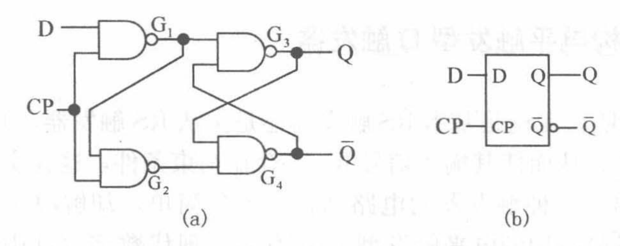


图 5-8 电平触发型D触发器结构与逻辑符号

对于之前出现的RS触发器的特征方程式（5-1），当CP=1时，将S=D，R=$\overline{D}$代入此钟控RS触发器的特性方程，即可得到与式（5-4）相同的D触发器的特性方程：

$$\begin{cases} Q^{n+1} = S + \overline{R}Q^n \\ RS = 0 \end{cases} \Rightarrow Q^{n+1} = D \tag{5-5}$$

显然，图5-8和图5-7的触发器的功能相同，都是电平触发型D触发器，在时钟脉冲的作用下，D触发器的新状态仅取决于输入信号D，而与原状态无关，因此图5-8的D触发器的真值表与表5-4相同，且拥有相同的时序波形图（图5-7（c））。此外，通常比较习惯用CLK来表示CP和ENA，所以图5-8（b）和图5-7（b）的逻辑符号是相同的。

传统设计中常见的锁存器通用集成电路的型号有74LS373、74LS75和74LS573等，这些电平触发型D触发器在计算机中常被用来锁存地址信号，称为地址锁存器。

5.3.3 维持-阻塞边沿触发型D触发器

电平触发型D触发器也有缺点，就是当CP=1期间，若数据输入端D有干扰电平，或错误的数据，它们将直接通过触发器出现在输出端Q上，即这时D触发器十分容易受到输入电平干扰，其状态是不确定的。这对于许多设备是不允许的。

边沿触发型D触发器的出现就是针对克服这种缺点的。此类触发器的特点是，在时钟的上升沿或下降沿时刻改变输出状态，且只在边沿前一瞬间，D端的输入信号是有效的。图5-9所示的电路就是比较经典的边沿触发型D触发器。

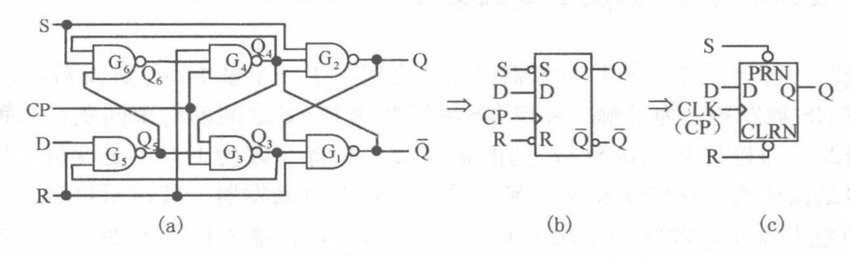


图 5-9 边沿触发型D触发器的结构与逻辑符号

图 5-9 所示的边沿触发型 D 触发器也称为维持-阻塞边沿 D 触发器，它由 6 个与非门组成，其中 G_1 和 G_2 组成基本 RS 触发器；G_3 和 G_4 组成时钟控制电路；G_5 和 G_6 组成数据输入电路。此触发器的工作原理简述如下：

电路中的 S 和 R 接至基本 RS 触发器的输入端，它们分别是预置和清 0 端，低电平有效。当 S=0 且 R=1 时，不论输入端 D 为何种状态，都会使 Q=1，$\overline{Q}$=0，即触发器被强制置 1；当 S=1 且 R=0 时，触发器的状态为 0。这里的 S 和 R 通常又称为直接置 1 和置 0 端。在分析其工作过程前，首先假设 S 和 R 均已加入了高电平，不再考虑其对电路工作的影响。触发器的工作过程如下：

(1) 当 CP=0 时，与非门 G_3 和 G_4 被封锁，其输出 $Q_3=Q_4=1$，触发器的状态不变。同时，由于 Q_3 至 Q_5 和 Q_4 至 Q_6 的反馈信号将 Q_5 和 Q_6 这两个门打开了，因此可接收输入信号 D。这时 $Q_5=\overline{D}$，$Q_6=\overline{Q_5}=D$。

(2) 当 CP 由 0 变 1 时，触发器发生翻转。与非门 G_3 和 G_4 被打开，它们的输出，即 Q_3 和 Q_4 的状态，由 G_5 和 G_6 的输出状态决定。$Q_3=\overline{Q_5}=D$，$\overline{Q_4}=\overline{Q_6}=\overline{D}$。于是由基本 RS 触发器的逻辑功能可知，Q=D。

(3) 当触发器翻转后，在 CP=1 时输入信号 D 被封锁。这是因为 G_3 和 G_4 打开后，它们的输出 Q_3 和 Q_4 的状态是相反的，即其中必有一个是 0。

若 Q_3 为 0，则经 G_3 输出至 G_5 输入的反馈线将 G_5 封锁，即封锁了 D 通往基本 RS 触发器的路径；该反馈线起到了使触发器维持在 0 状态并阻止触发器变为 1 状态的作用，故该反馈线称为置 0 维持线，或置 1 阻塞线。

若 Q_4 为 0，则将 G_3 和 G_6 封锁，D 端通往基本 RS 触发器的路径也被封锁。于是 Q_4 输出端至 G_6 反馈线起到使触发器维持在 1 状态的作用，称作置 1 维持线；Q_4 输出至 G_3 输入的反馈线起到阻止触发器置 0 的作用，称为置 0 阻塞线。因此，该触发器常称为维持-阻塞触发器。

总之，该触发器是在 CP 正跳沿前接收输入信号，正跳沿时触发翻转，正跳沿后输入即被封锁，三步都是在正跳沿处完成的，所以有边沿触发器之称。与主从触发器相比，同工艺的边沿触发器具有更强的抗干扰能力和更高的工作速度。

图 5-9（b）所示的为边沿 D 触发器的传统逻辑符号。该逻辑符号是指上升沿触发型 D 触发器，这里，符号“^”表示时钟 CP 为边沿触发响应型，以区分于电平触发型；若再加小圆圈“o”后，则表示下降沿触发。实用中，通常都是上升沿触发型。

图 5-9（c）所示的是现代数字技术中最常用的边沿 D 触发器的逻辑符号。图中 PRN、CLRN 分别为异步置 1 端和异步置 0 端（异步复位端），它们分别对应传统触发器的 S 和 R 端。

边沿 D 触发器的特性方程表达式及真值表仍与电平触发 D 触发器的特性方程相同，只是输出状态发生变化的时刻不同。边沿 D 触发器在时钟脉冲的上升沿或下降沿时刻，将上升沿或下降沿前一瞬间的输入数据 D 传输到输出端。

常用的集成电路边沿型 D 触发器的型号是 74LS74，其内部包括两个相同的上升边沿触发型 D 触发器，逻辑端口结构图如图 5-10 所示，内部的单个 D 触发器端口结构如图 5-11 所示。

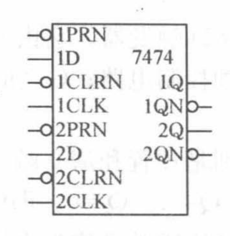


图 5-10 74LS74 端口逻辑图

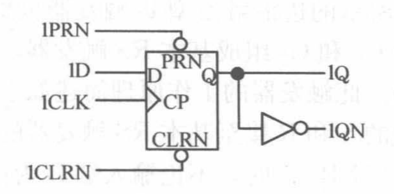


图 5-11 7474 内部的边沿 D 触发器

前面已经提到，边沿型 D 触发器是现代数字逻辑系统中的基本时序元件。触发器的应用不仅可以对周期波形进行分频，而且还可以实现计数、数据存储等功能。边沿型 D 触发器由于功能齐全，结构简单，控制方便，容易实现自动化设计，因此在现代数字系统中应用最为广泛。

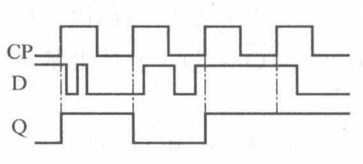


图 5-12 例 5-1 波形图

【例 5-1】 图 5-12 是上升沿触发型 D 触发器的输入信号和时钟脉冲波形，设触发器的初始状态为 0，确定输出信号 Q 的波形。

解：把握边沿触发型 D 触发器工作特性的关键是，确认每个时钟脉冲 CP 上升沿之后的输出状态等于该上升沿前一瞬间 D 信号的状态，此状态将保持到下一个时钟脉冲 CP 上升沿到来时。由此可画出输出 Q 的波形如图 5-12 所示。

【例 5-2】 图 5-13 是两个边沿 D 触发器构成的电路图，设触发器的初始状态 $Q_1Q_0=00$，试确定 Q_0 及 Q_1 在时钟脉冲作用下的波形（参考图 5-14 的输入波形）。

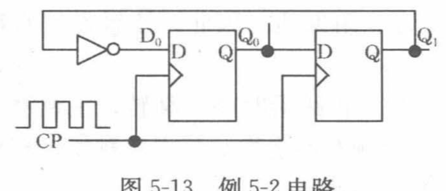


图 5-13 例 5-2 电路

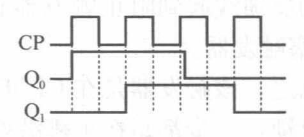


图 5-14 例 5-2 波形图

解：由于两个 D 触发器的输入信号分别为另一个 D 触发器的输出，因此在确定它们的输出端波形时，应分段交替画出 Q_0 及 Q_1 的波形，如图 5-14 所示。

第 1 个 CP 脉冲到来时，由于初始状态 $Q_1Q_0=00$，$D_0=1$，$D_1=0$，因此 $Q_0=1$，$Q_1=0$；

第 2 个 CP 脉冲到来时，由于现态 $Q_1Q_0=01$，$D_0=1$，$D_1=1$，因此 $Q_0=1$，$Q_1=1$；

第 3 个 CP 脉冲到来时，现态 $Q_1Q_0=11$，$D_0=0$，$D_1=1$，因此 $Q_0=0$，$Q_1=1$；

第 4 个 CP 脉冲到来时，现态 $Q_1Q_0=10$，$D_0=0$，$D_1=0$，因此 $Q_0=0$，$Q_1=0$。

5.3.4 由 CMOS 传输门构成的各类 D 触发器

在现代 CMOS 集成电路中，大多数触发器是由 CMOS 传输门构成的。

1. 电平触发型D触发器

以CMOS传输门构成的电平触发型D触发器（D-Latch，D锁存器）的内部电路结构如图5-15所示。图中，当CP=1时，传输门TG1的c端为1，cn端为0，TG1处于导通状态；而传输门TG2的c端为0，cn端为1，处于输出高阻状态。因此，D能通过TG1和反相器INV1，使得$\overline{Q}=\overline{D}$，而其输出通过反相器INV2，使得Q=D，但INV2的输出却不能通过TG2去改变INV1的输入。

当CP=0时，传输门TG1处于输出高阻状态，D信号无法传递到INV1的输入上，而TG2处于导通状态，反相器INV1与INV2就构成了一个反馈的回环。显然，该回环可以寄存0或者1两种状态，而当前回环中被锁存的到底是0还是1，完全取决于构成回环的瞬间Q与$\overline{Q}$的取值。进一步分析图5-15可以发现，TG1与TG2构成了类似于2选1多路选择器的结构：当CP=1时，反相器INV1的输入被选择为D，而当CP=0时，INV2的输入被选择为输出Q的反馈。从CP=1到CP=0的瞬间，反馈的回环构成了。也就是说，CP=0而输出Q值，完全取决于当CP从1变到0的最后时刻Q的取值。

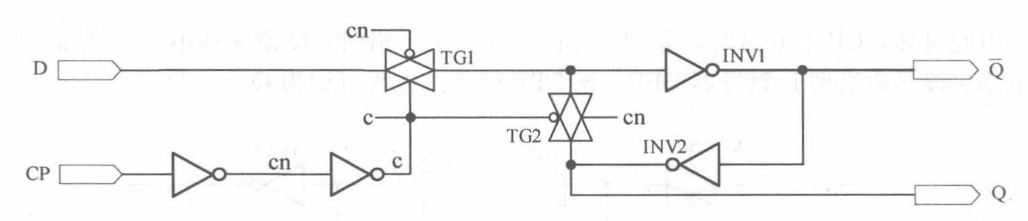


图5-15 CMOS传输门构成的电平触发型D触发器

归纳起来，图5-15所示电路的功能是：当CP=1时，Q=D（直通状态）；而当CP=0时，Q保持原有状态不变（锁存状态）。显然，此电路即为电平型D触发器。

2. 带清0和置位控制端的电平型D触发器

若将图5-15所示电路稍作修改，把INV1和INV2两个反相器换成二输入与非门，如图5-16所示的电路形式，即可改变CP=0时反馈回环的当前状态。在SETN=0，CLRN=1时，可使输出信号Q=1；而SETN=1，CLRN=0时，可使Q=0。正好构成带清0和置位端的电平型D触发器。

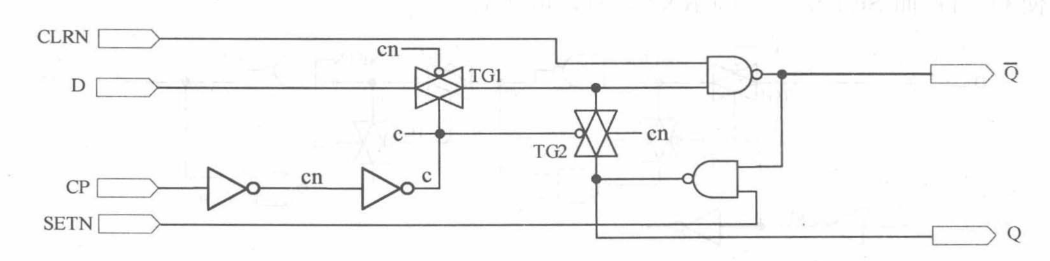


图5-16 CMOS传输门构成的带清0和置位端的电平触发型D触发器

3. 边沿触发型D触发器

利用CMOS传输门也能构成边沿触发型D触发器（DFF），其电路结构如图5-17所示。观察此图可以发现，其电路结构是由两个类似于图5-15所示电路结构的电平型D触发器构成的。图5-17所示电路中传输门TG1、TG2和反相器INV1、INV2构成一个电平型D触发器，而传输门TG3、TG4和反相器INV3、INV4构成了另一个电平型D触发器。按照位置的不同，以下分别称前级或后级电平型D触发器。

当CP=0时，前级电平型D触发器处于直通状态，$q1=\overline{D}$，而后级处于锁存状态；当CP=1时，后级电平型D触发器处于直通状态，$Q=\overline{q1}$，而前级处于锁存状态，保存的是从CP=0到CP=1前D的输入值，即CP上升沿时D的输入值。

假定输入的时钟信号CP先为0后为1，那么当CP=0时，前级输出$q1=\overline{D}$，后级输出Q保持原值不变；当CP从0变为1的瞬间，前级保持q1输出不变，即为$\overline{D}$（D为CP=0最后时刻的前级的输入值），后级直通，$Q=\overline{q1}$，而这时的$q1=\overline{D}$，即Q=D（D为CP=0最后时刻的前级的输入值）；当CP=1时，前级保存上升沿时的输出，后级直通输出前级的值（反相）。

由此可知，CP上升沿时，Q=D，而CP不为上升沿时，Q值保持不变，即图5-17所示电路应该是边沿型D触发器DFF。该DFF的状态方程可记为$Q^{n+1}=D$。

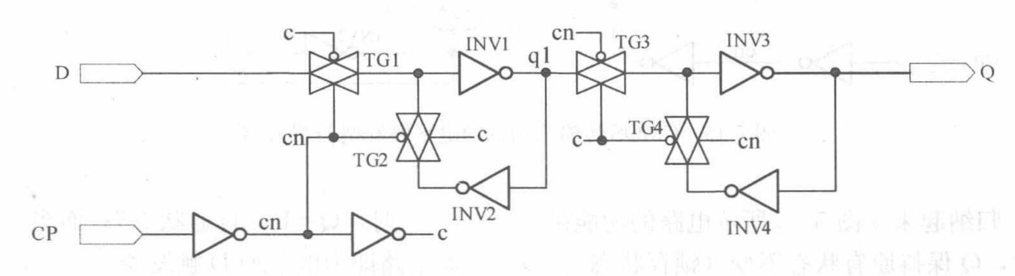


图5-17　CMOS传输门构成边沿型触发型D触发器（DFF）

4. 带清0和置位控制端的边沿触发型D触发器

与图5-16电路结构类似，可以通过增加与非门得到带清0和置位端的DFF的CMOS传输门结构图，其电路如图5-18所示。当CP不为上升沿，且SETN=0，CLRN=1时，可使Q=1；而SETN=1，CLRN=0时，可使Q=0。

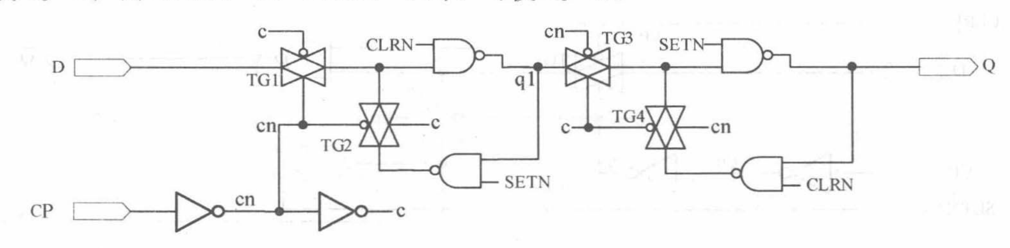


图5-18　CMOS传输门构成的带清0和置位端的边沿型D触发器

5.4 JK触发器

与D触发器相同，JK触发器也具有置0、置1、保持、翻转和边沿触发等功能，而且也能灵活地构成诸如D触发器、T触发器等其他类型的触发器。相比而言，在各类集成触发器中，JK触发器功能最为齐全，并有很强的通用性。尽管在实用的广泛性方面略逊于D触发器，但仍然值得对其结构和功能作一些认知和探讨。

5.4.1 主从JK触发器

若对主从RS触发器通过引入如下反馈，可获得主从JK触发器：

$$S = J\overline{Q^n} \qquad R = KQ^n$$

则此时输入S、R自动满足约束条件，且主触发器只发生一次翻转。获得的逻辑电路如图5-19（a）所示，即主从JK触发器电路；图5-19（b）是其逻辑符号。

将输入S、R的表达式代入主从RS触发器的特性方程（5-3），即可得到主从JK触发器的特性方程：

$$\begin{aligned} Q^{n+1} &= S + \overline{R}Q^n = J\overline{Q^n} + \overline{KQ^n}Q^n \\ &= J\overline{Q^n} + \overline{K}Q^n \end{aligned} \tag{5-6}$$

上式仅在CP的下降沿到来时有效。注意上式中Q^{n+1}为CP下降沿之后的状态，Q^n为CP下降沿之前的状态，J、K信号为CP＝1期间的值。波形图5-19（c）反映了主从触发器的工作特性（注意波形图中，输入与输出信号间有延迟现象）。

由特性方程（5-6）可得到主从JK触发器的状态转换真值表，如表5-5所示。

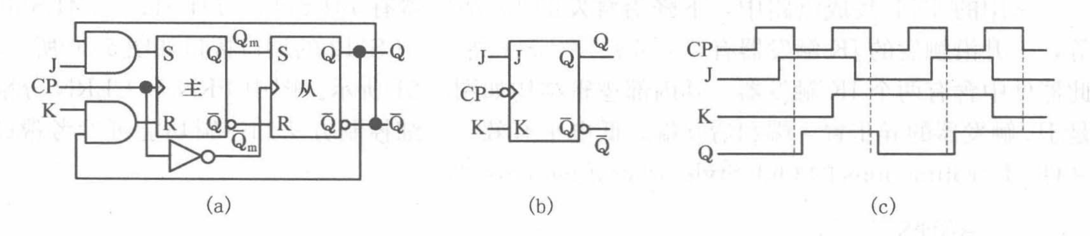


图5-19 主从JK触发器的内部电图路及其逻辑符号

表5-5 主从JK触发器状态转换真值表（CP下降沿时）

J	K	Q^n	Q^{n+1}	功能	
0	0	0	0	$Q^{n+1}=Q^n$	保持
0	0	1	1		
0	1	0	0	$Q^{n+1}=0$	置0
0	1	1	0		
1	0	0	1	$Q^{n+1}=1$	置1
1	0	1	1		
1	1	0	1	$Q^{n+1}=\overline{Q^n}$	翻转
1	1	1	0		

根据主从 JK 触发器的波形图（如图 5-19（c）所示）可知：

在第 1 个 CP 高电平期间，J=1，K=0，CP 下降沿后，Q^{n+1}为 1，触发器被置 1；

在第 2 个 CP 高电平期间，J=0，K=1，CP 下降沿后，Q^{n+1}置为 0；

在第 3 个 CP 高电平期间，J=1，K=1，CP 下降沿后，Q^{n+1}翻转为 1；

在第 4 个 CP 高电平期间，J=0，K=0，Q^{n+1}保持不变，如此等等。

注意主从 JK 触发器的主触发器在一个时钟周期中最多只能翻转一次，这种现象称为主从 JK 触发器的“一次翻转”。如果在 CP=1 期间，输入信号 J、K 发生了变化，在时钟脉冲 CP 的下降沿到来时，主从 JK 触发器的输出就有可能与式（5-6）不一致，这也正是由于主从 JK 触发器的“一次翻转”特性造成的。

5.4.2 边沿触发型 JK 触发器

边沿 JK 触发器弥补了主从 JK 触发器的“一次翻转”问题，因此更为实用。边沿 JK 触发器最方便的获取方法是利用 D 触发器来转换，或用 CMOS 传输门来构建。

边沿 JK 触发器的特点如下：

（1）边沿 JK 触发器在 CP 信号的边沿到来时产生翻转，在 CP 有效边沿前瞬间的 J、K 输入信号为有效输入信号。

（2）对于主从 JK 触发器，在 CP=1 的全部时间内，J、K 输入信号均为有效输入信号。故与主从 JK 触发器相比，边沿 JK 触发器大大减少了干扰信号可能作用的时间，从而增强了抗干扰能力。

（3）边沿 JK 触发器的真值表、特性方程与主从 JK 触发器完全相同。

（4）边沿 JK 触发器无“一次翻转”问题。

常用的 TTL 集成电路中，下降沿触发的 JK 触发器有 74LS112、74LS113、74LS114 等，上升沿触发的 JK 触发器有 74LS73、74LS76 等。74LS112 的端口接口如图 5-20 所示，此器件中含有两个 JK 触发器，其内部逻辑结构如图 5-21 所示。图中 PRN 和 CLRN 分别是 JK 触发器的异步置 1 端和清 0 端，低电平有效。详细控制方法和逻辑功能可参考帮助文件 Macrofunctions 的 Old_Style Macrofunctions 项。

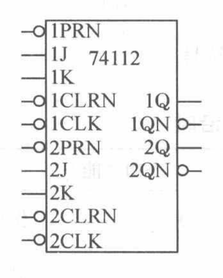


图 5-20 下降沿 JK 触发器

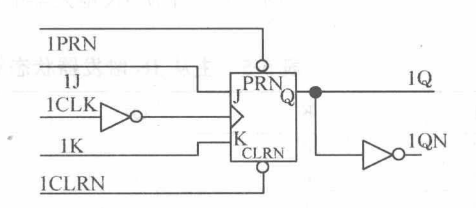


图 5-21 下降沿触发型 JK 触发器 74112 部分内部结构

【例 5-3】设上升沿 JK 触发器电路如图 5-22 所示，其初态为 0，输入信号波形如图 5-23 所示，试画出它的输出波形。

解：在画 JK 触发器 Q 的输出波形时，可根据 JK 状态转换真值表，在 CLK 脉冲上升

沿到来时发生翻转，而在CLK高电平和低电平期间，状态保持不变。此题中要特别注意的是此触发器的同步置0、置1端（RD、SD）的操作受时钟CLK的控制。

其输出波形如图5-23所示。

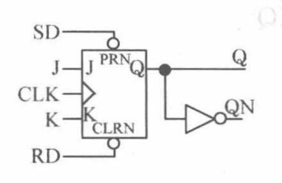


图5-22 例5-3电路图

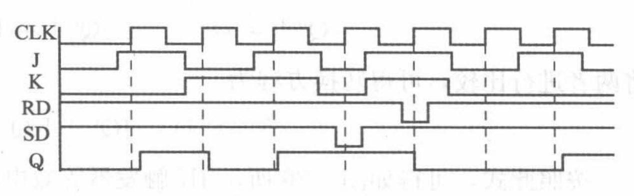


图5-23 例5-3输入输出仿真波形图

【例5-4】图5-24给出了由两个边沿JK触发器连接而成的逻辑电路，设两个触发器的初始状态都是0状态，试确定输出端Q_1、Q_0的波形，并写出由这些波形所表示的二进制序列。

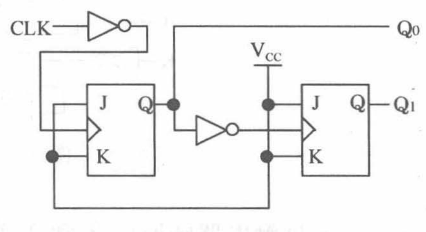


图5-24 例5-4逻辑电路图

解：根据边沿JK触发器的特点，可得到Q_1、Q_0的输出波形如图5-25所示。若将Q_1、Q_0的时序进行排列，即为00，01，10，11，分别对应于0，1，2，3。

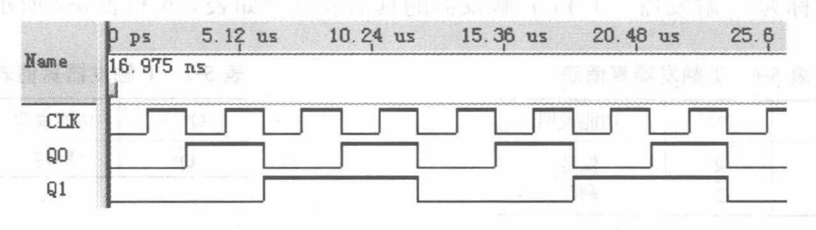


图5-25 例5-4输出波形

由此可见，这个二进制序列每4个时钟脉冲重复一次，然后返回0重新开始该序列，此序列相当于对时钟脉冲进行了计数。

5.5 触发器间的转换

不同类型的触发器的结构与特性虽然不同，但由时钟控制的具有寄存数据时序特性的本质是相同的，因此触发器之间通常是可以互为转换的，这给实际应用带来了方便。以下介绍触发器间的转换方法。互为转换是根据已有触发器和待求触发器的特性方程相等的原则，求出已有触发器的输入信号与待求触发器之间的转换逻辑关系。

5.5.1 D触发器转换为JK、T和T′触发器

D触发器在目前是使用最普遍的基本时序单元，这里重点介绍它向其他触发器转换的实例。

(1) D 触发器转换成 JK 触发器。在触发器中，D 触发器和 JK 触发器具有较完善的功能，最常用的集成触发器大多数也是 D 或 JK 触发器，而且它们之间可以互相转换。

转换方法是，首先写出 D 触发器和 JK 触发器的特性方程：

$$Q^{n+1} = D \qquad Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$$

将两者进行比较，可得转换方程为

$$D = J\overline{Q^n} + \overline{K}Q^n$$

按照此式，可得如图 5-26 所示 JK 触发器等效电路。

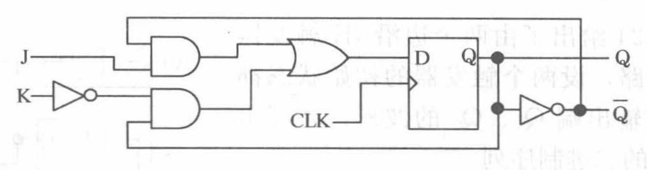


图 5-26　用 D 触发器构成的 JK 触发器

(2) D 触发器转换成 T 和 T′触发器。在 CP 时钟脉冲控制下，根据输入信号 T 取值的不同，只具有保持和翻转功能的电路，称为 T 触发器；凡是每来一个时钟脉冲就翻转一次的电路，称为 T′触发器。T 和 T′触发器的真值表分别如表 5-6 和表 5-7 所示。

表 5-6　T 触发器真值表

T	Q^{n+1}	功能说明
0	Q^n	保持
1	$\overline{Q^n}$	翻转

表 5-7　T′触发器真值表

Q^{n+1}	功能说明
$\overline{Q^n}$	翻转

在 PLD 器件内或实际的集成电路的设计库中，没有单独的 T、T′触发器标准单元。如果有必要时，它们通常由其他触发器（主要是 D 触发器）转换而来。在此首先讨论 D 触发器转换为 T 触发器，采用与 D 触发器构成 JK 触发器相同的方法。

根据 T 触发器的特性方程 $Q^{n+1} = T\overline{Q^n} + \overline{T}Q^n = T \oplus Q^n$，与 D 触发器的特性方程 $Q^{n+1} = D$ 相比较可得

$$D = T \oplus Q^n$$

按照上式，就可以将 D 触发器转换成 T 触发器了。电路如图 5-27 所示。同样将 D 触发器转换为 T′触发器时，根据 T′触发器的特性方程 $Q^{n+1} = \overline{Q^n}$，可得 $D = \overline{Q^n}$，由此可得如图 5-28 所示的 T′触发器电路。

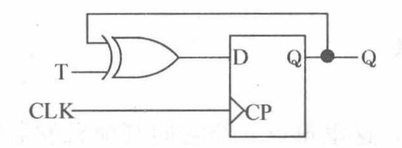


图 5-27　用 D 触发器构成的 T 触发器

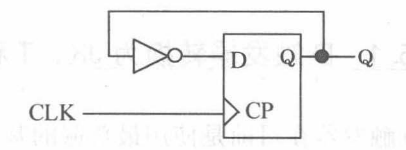


图 5-28　用 D 触发器构成的 T′触发器

5.5.2 JK 触发器转换为 D 触发器

写出待求D触发器的特性方程，并进行变换，使之形式与已有的JK触发器的特性方程一致：

$$Q^{n+1} = D = DQ^n + D\overline{Q^n}$$

再与JK触发器的特性方程 $Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$ 进行比较，可得到：$J=D$，$K=\overline{D}$，由此可得D触发器的电路如图5-29所示。JK触发器转换成T和T′触发器的方法与此类同。

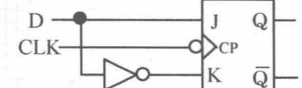


图 5-29 JK触发器构成的D触发器

5.6 含触发器的PLD的结构与原理

在第4章的最后一节曾介绍了有关可编程逻辑器件PLD的结构和工作原理，但由于之前的内容尚未接触到触发器的概念，因此主要限制于基于组合电路的可编程器件结构的介绍。本节将介绍目前常用的PLD器件的结构原理，包括通用阵列逻辑器件GAL、复杂可编程逻辑器件CPLD和现场可编程门阵列FPGA。它们的可编程结构包含可编程的组合电路结构和基于触发器的时序电路可编程结构。对于这些器件的了解有助于更深入地理解和更好地掌握现代逻辑电路的自动设计技术。

5.6.1 通用可编程逻辑器件 GAL

早在1985年，美国Lattice公司在PAL的基础上，设计出了通用阵列逻辑器件GAL。首次在PLD上采用了EEPROM工艺，使得GAL具有电可擦除重复编程的特点，彻底解决了熔丝型可编程器件的一次可编程问题。GAL在“与-或”阵列结构上沿用了PAL的与阵列可编程、或阵列固定的结构，但对早期的PAL的输出I/O结构进行了较大的改进，在GAL的输出部分增加了输出逻辑宏单元OLMC（Output Logic Macro Cell）。图5-30是型号为GAL16V8器件的结构图，其基本结构与图4-50的PAL16V8十分相似。

GAL的OLMC单元设有多种组态，可配置成（即可编程成）专用组合输出、专用输入、组合输出双向口、寄存器输出、寄存器输出双向口等，为逻辑电路设计提供了极大的灵活性。由于具有结构重构和输出端的任何功能均可移到另一输出引脚上的功能，在一定程度上，简化了电路板的布局布线，使系统的可靠性进一步地提高。

如图5-30所示，GAL的基本结构包括：

- 8个输入缓冲器和8个输出反馈/输入缓冲器。
- 8个逻辑宏单元OLMC、8个三态缓冲器，每个OLMC对应一个I/O引脚。
- 由8×8个与门（32输入与门）构成的与阵列，共可形成64个乘积项，每个与门有32个输入项，由8个输入的原变量、反变量和8个反馈信号的原变量、反变量组成。所以其可编程与阵列共有2048个可编程单元。
- 系统时钟CLK和三态输出选通信号的输入缓冲器。

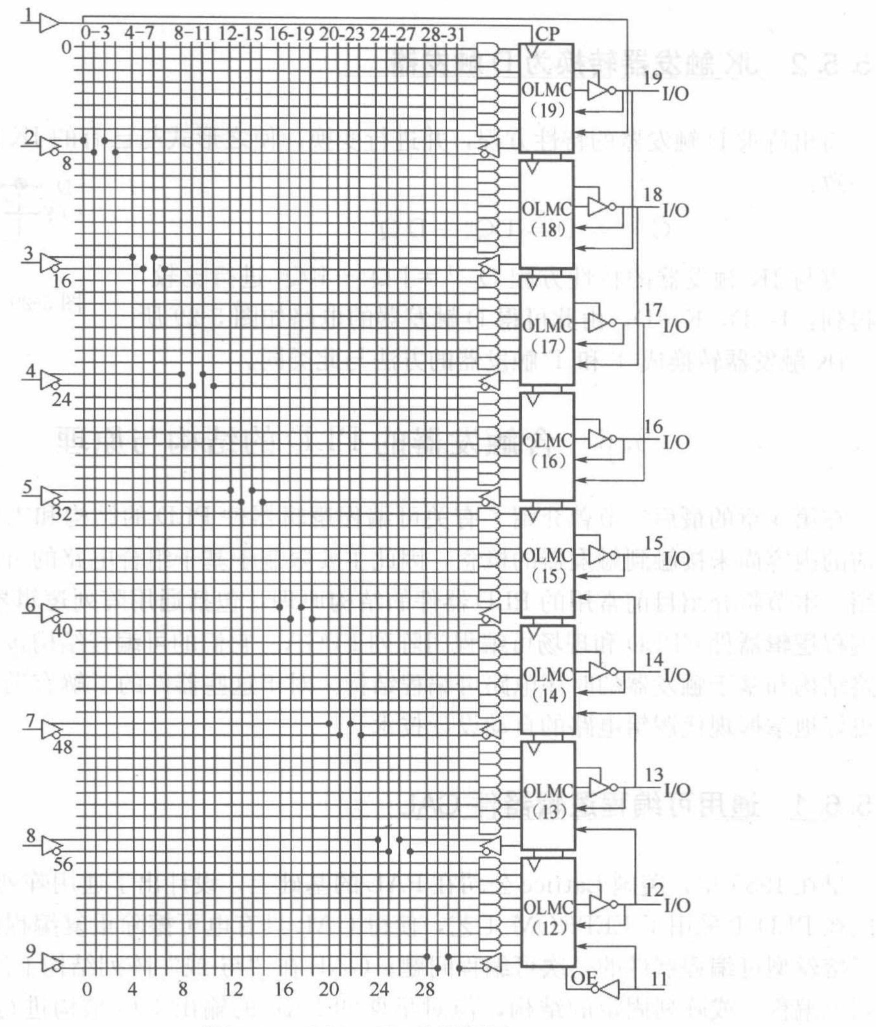


图 5-30　GAL16V8 的逻辑图

GAL 的输出逻辑宏单元 OLMC（图 5-31）中有 4 个多路选择器，通过不同的选择方

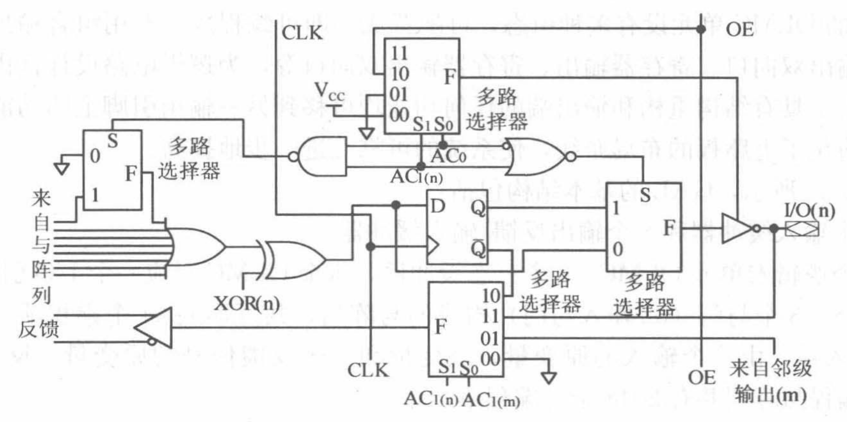


图 5-31　逻辑宏单元 OLMC 的逻辑结构图

式可以产生多种输出结构，分别属于3种模式。一旦确定了某种模式，所有的OLMC都将工作在同一种模式下。3种输出模式如下。

(1) 寄存器模式。在寄存器模式下，OLMC有如下两种输出结构：

① 寄存器输出结构（图5-32）。异或门输出经D触发器至三态门，触发器的时钟端连公共CLK引脚（图5-30的第1脚）、三态门的使能端OE连公共OE引脚（图5-30的第11脚），信号反馈来自D触发器。

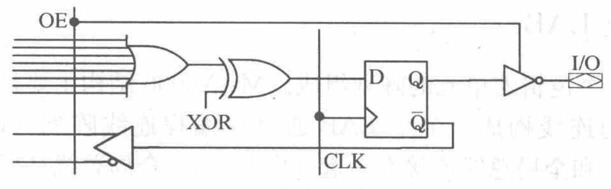


图5-32 寄存器输出结构

② 寄存器模式组合输出双向口结构。输出三态门受控，输出反馈至本单元，组合逻辑输出，无触发器。

(2) 复合模式。在复合模式下，OLMC有如下两种结构：

① 组合输出双向口结构。大致与寄存器模式下组合输出双向口结构相同，区别是引脚CLK、OE在寄存器模式下为专用公共引脚，不可作它用。

② 组合输出结构。无反馈，其他与组合输出双向口结构相同。

(3) 简单模式。此模式下，构成含反馈输入或反馈输出型纯组合电路。

5.6.2 复杂可编程逻辑器件CPLD

早期的CPLD从GAL的结构扩展而来，但针对GAL的缺点进行了改进，如Lattice公司的ispLSI1032器件等。在流行的CPLD中，Altera公司的MAX3000A/S系列器件的结构和功能具有一定典型性，在这里以此为例简要介绍CPLD的结构和工作原理。

MAX3000系列器件包含32到256个宏单元，其单个宏单元结构如图5-33所示。每

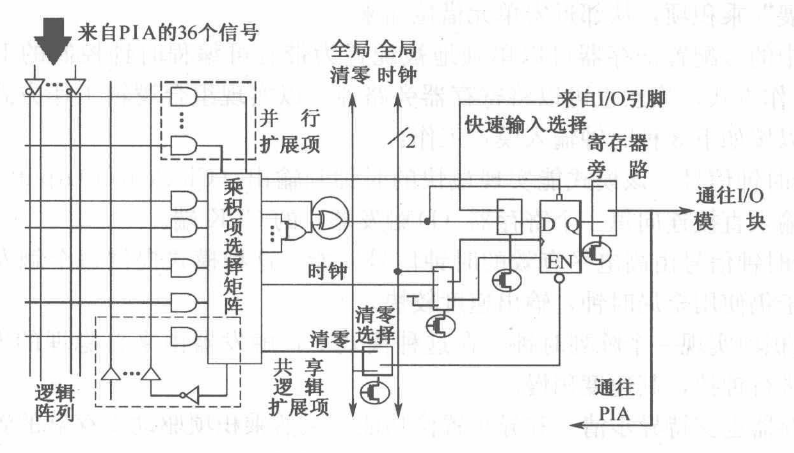


图5-33 MAX3000系列的单个宏单元结构

16个宏单元组成一个逻辑阵列块LAB（Logic Array Block）。每个宏单元含有一个可编程的“与”阵列和固定的“或”阵列，以及一个可配置寄存器；每个宏单元共享扩展乘积项和高速并联扩展乘积项，它们可向每个宏单元提供多达32个乘积项，从而可实现复杂的逻辑函数。显然，图5-33的结构与GAL的OLMC（图5-31）有很大的相似性。

MAX3000结构中包含5个主要部分，即逻辑阵列块、宏单元、扩展乘积项（共享和并联）、可编程连线阵列、I/O控制块。

1. 逻辑阵列块LAB

一个LAB由16个逻辑宏单元的阵列组成。MAX3000结构主要是由多个LAB组成的阵列以及它们之间的连线构成。多个LAB通过可编程连线阵列（PIA，Programmable Interconnect Array）和全局总线连接在一起（图5-34），全局总线从所有的专用输入、I/O脚和逻辑宏单元馈入信号。每个LAB的输入信号来自3个方面：

- 作为通用逻辑输入的PIA的36个信号。
- 全局控制信号，用于寄存器辅助功能。
- 从I/O引脚到寄存器的直接输入通道。

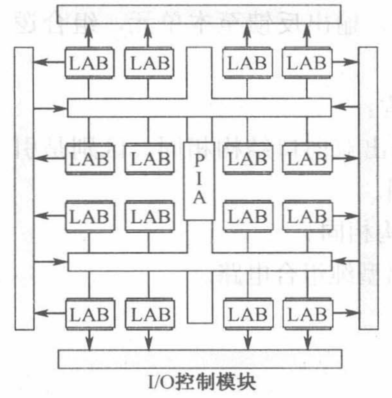


图5-34　MAX3128S的结构

2. 逻辑宏单元

MAX3000系列中的逻辑宏单元（类似于GAL的OLMC）由3个功能块组成：逻辑阵列、乘积项选择矩阵和可编程寄存器，它们可以被单独地配置为时序逻辑和组合逻辑工作方式。其中逻辑阵列实现组合逻辑，可以给每个宏单元提供5个乘积项。“乘积项选择矩阵”分配这些乘积项作为到或门和异或门的主要逻辑输入，以实现组合逻辑函数；或者把这些乘积项作为宏单元中寄存器的辅助输入：清0（Clear）、置位（Preset）、时钟（Clock）和时钟使能控制（Clock Enable）。

每个宏单元中有一个“共享扩展”乘积项经非门后回馈到逻辑阵列中，宏单元中还存在“并行扩展”乘积项，从邻近宏单元借位而来。

宏单元中的可配置寄存器可以单独地被配置为带有可编程时钟控制的D、T、JK或RS触发器工作方式；当然也可以将寄存器旁路掉，以实现组合逻辑工作方式。每个可编程寄存器可以按如下3种时钟输入模式工作：

- 全局时钟信号。该模式能实现最快的时钟到输出（Clock to Output）的性能，这时全局时钟输入直接连向每一个寄存器（D触发器）的CLK端。
- 全局时钟信号由高电平有效的时钟信号使能。这种模式提供每个触发器的时钟使能信号，由于仍使用全局时钟，输出速度较快。
- 用乘积项实现一个阵列时钟。在这种模式下，触发器由来自隐埋的宏单元或I/O引脚的信号进行钟控，其速度稍慢。

每个寄存器也支持异步清0和异步置位功能。虽然乘积项驱动寄存器的置位和复位信号是高电平有效，但在逻辑阵列中将信号取反可得到低电平有效的效果。此外，每一个寄

存器的复位端可以由低电平有效的全局复位专用引脚 GCLRn 信号来驱动。

5.6.3 现场可编程门阵列 FPGA

FPGA 是大规模可编程逻辑器件除 CPLD 外的另一大类 PLD 器件，也是目前发展最快、逻辑规模最大、使用范围最广的可编程逻辑器件。特别是随着适用领域的不断扩大，器件性能和功能的不断改善，FPGA 的结构与功能模块的构成也在不断改变。因此，以下将简要介绍 FPGA 的一些典型硬件结构和可编程原理。

注意：读者会在此遇到一些在前面章节中未曾接触过的概念，如存储器、RAM、SRAM、计数器、移位寄存器等。这没有关系，可以暂时跳过去，尽量把注意力放在对 FPGA 基本结构和可编程原理的理解上，至于这些新概念都将在以后的章节中介绍。提前学习 FPGA 的目的就是为了在接下去的章节所给出的实验中更好地使用 Quartus II 完成各项实验要求。对于此节暂时搁置的概念可以在学完相关章节后再作回顾。

1. 查找表逻辑结构

前面提到的诸如 GAL、CPLD 之类可编程逻辑器件，都是基于乘积项的可编程结构的，即由可编程的与阵列和固定的或项组成组合逻辑。而在本节中将要介绍的 FPGA，使用了另一种可编程逻辑的形成方法，即可编程的查找表（Look Up Table，LUT）结构，其中 LUT 是组合逻辑可编程的最小逻辑构成单元。大部分 FPGA 采用了基于 SRAM（一种存储器）的查找表逻辑形成结构，就是用 SRAM 来构成逻辑函数发生器。

一个 N 输入查找表（LUT）可以实现 N 个输入变量的任何逻辑功能，图 5-35 是 4 输入 LUT，其内部结构模型如图 5-36 所示，是一个最小的 LUT 单元。图 5-36 中输出信号与输入信号 A、B、C、D 之间的逻辑关系取决于 16 个单元的 RAM（一种存储器）中每一个单元预先存放的二进制数据。输入信号 A、B、C、D 的实际功能是通过输入的高低电平控制对应的多路选择器。例如输入信号 C 控制 2 个多路选择器；信号 B 控制 4 个多路选择器。而 16×1 RAM 中的预先存入的数据来自计算机根据系统设计者描述的逻辑电路所对

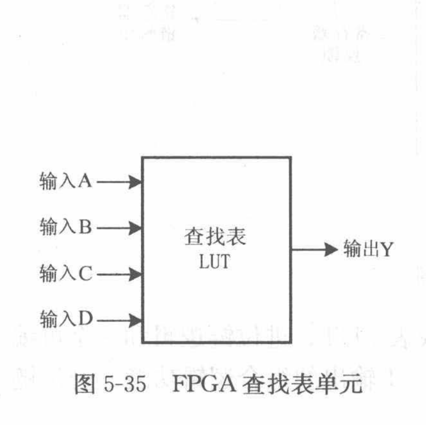


图 5-35 FPGA 查找表单元

图 5-36 FPGA 查找表单元内部结构

应的逻辑关系，“翻译”而来的二进制数据。

对于一个N端输入的查找表，需要SRAM存储N个输入构成的真值表，需要用2的N次幂个位的SRAM单元。显然N不可能很大，否则LUT的利用率很低，输入多于N个的逻辑函数，必须用多个查找表分开实现。

Xilinx公司的XC4000系列、Spartan系列，Altera公司的FLEX10K、ACEX、APEX、Cyclone等系列都采用了SRAM查找表构成，是典型的FPGA器件。

2. Cyclone系列FPGA器件的基本结构

本书中多数示例和实验是基于Altera公司的Cyclone系列的FPGA的，也包括Cyclone的一些后续系列，如Cyclone II、Cyclone III、Cyclone IV等系列。

Cyclone系列FPGA器件是Altera公司开发的一款低成本、高性价比的FPGA。它的结构和工作原理在FPGA器件中具有典型性，这里以此类器件为例，介绍FPGA的结构。Cyclone器件主要由逻辑阵列块（LAB）、嵌入式存储器块、I/O单元和EAB、M4K、PLL等嵌入式模块构成，在各个模块之间存在着丰富的互连线和时钟网络。

Cyclone器件的可编程资源主要来自逻辑阵列块LAB，而每个LAB都是由多个LE来构成。LE（Logic Element）即逻辑宏单元，是Cyclone FPGA器件的最基本的可编程单元，职能类似于GAL的OLMC。图5-37显示了Cyclone FPGA的LE单元的内部结构。

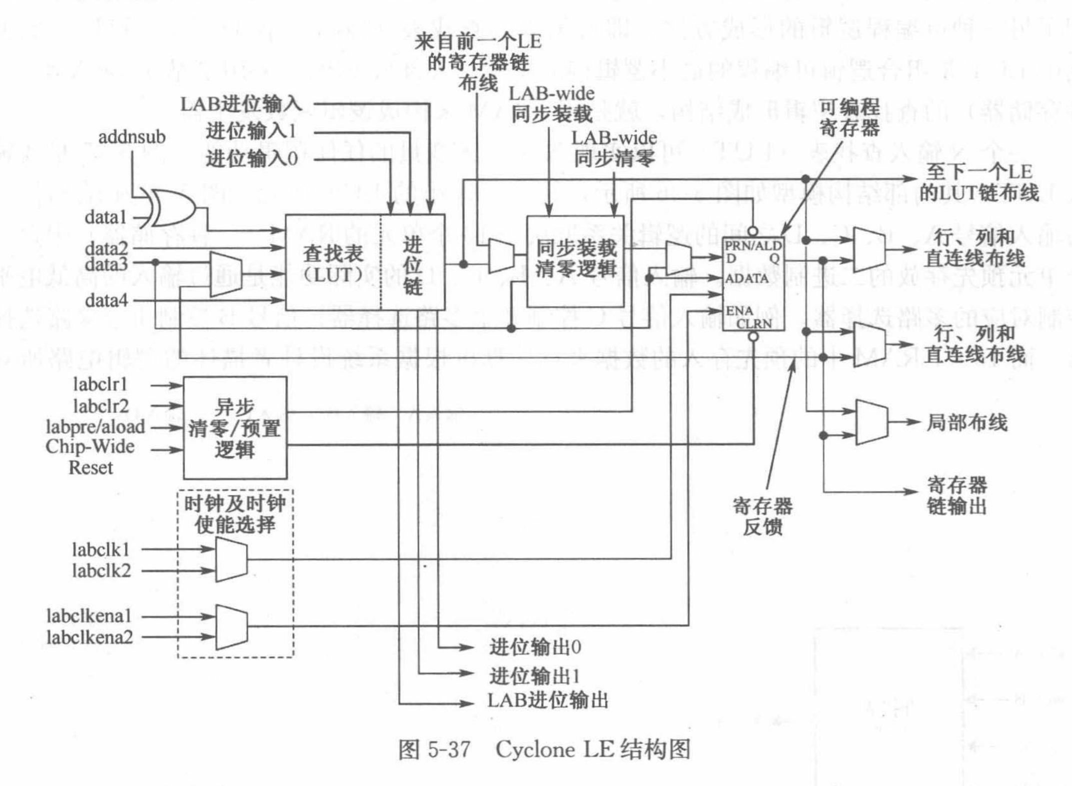


图5-37 Cyclone LE结构图

观察图5-37可以发现，LE主要由一个4输入的查找表LUT、进位链逻辑和一个可编程的寄存器构成。4输入的LUT可以完成所有的4输入、1输出的组合逻辑功能。进位链

逻辑带有进位选择，可以灵活地构成1位加法或者减法逻辑，并可以切换。每一个LE的输出都可以连接到局部布线、行列、LUT链、寄存器链等布线资源。

每个LE中的可编程寄存器可以被配置成D、T、JK和RS触发器模式。每个可编程寄存器具有同步数据装载、异步数据装载、时钟、时钟使能、清0和异步置位/复位控制信号。

LE中的时钟、时钟使能选择逻辑可以灵活配置寄存器的时钟以及时钟使能信号。对于只需完成组合逻辑实现的情况，可将该触发器旁路，LUT的输出可作为LE的输出。

LE有3个输出驱动内部互连，其中一个驱动局部互连，另两个驱动行或列的互连资源。LUT和寄存器的输出可以单独控制。可以实现在一个LE中，LUT驱动一个输出，而寄存器驱动另一个输出。因而在一个LE中的触发器和LUT能够用来完成不相关的功能，因此能够提高LE的资源利用率。

除上述的3个输出外，在一个逻辑阵列块中的LE，还可以通过LUT链和寄存器链进行互连。在同一个LAB中的LE通过LUT链级联在一起，可以实现宽输入（输入多于4个）的逻辑功能。在同一个LAB中的LE里的寄存器可以通过寄存器链级联在一起，构成一个移位寄存器，而LE中LUT资源可以单独实现组合逻辑功能。

3. Cyclone中的嵌入式模块

在Cyclone FPGA器件中含有许多功能强大、使用灵活的嵌入式模块。

嵌入式存储器（Embedded Memory）是最典型、应用最广的嵌入式模块之一。嵌入式存储器一般称为EAB（嵌入式阵列块）。Cyclone中的EAB称为M4K，由数十个M4K的存储器块构成。每个M4K存储器块具有很强的伸缩性，可以实现的功能包括：4096位RAM、200 MHz高速性能存储器、真正的双端口存储器、单个双端口存储器、单端口存储器、字节使能、校验位、移位寄存器、FIFO和ROM等，所有这些都可以通过编程设计组合并调用FPGA中已嵌入的大量EAB模块来构建。

在Cyclone中的嵌入式存储器可以通过多种连线与可编程资源实现连接，这大大增强了FPGA的性能，扩大了FPGA的应用范围。此外，在Cyclone FPGA中还含有1～4个嵌入式锁相环PLL，可以用来调整时钟信号的波形、频率和相位。

Cyclone支持多种I/O接口，符合多种I/O标准，可以支持差分的I/O标准，诸如LVDS（低压差分串行）和RSDS（去抖动差分信号），当然也支持普通单端的I/O标准，比如LVTTL（低压TTL电平）、LVCMOS、SSTL和PCI等，通过这些常用的端口与其他芯片沟通。

Cyclone器件可以支持最多129个通道的LVDS和RSDS。器件内的LVDS缓冲器可以支持最高达640兆比特每秒（Mbps）的数据传输速度。与单端的I/O标准相比，这些内置于Cyclone器件内部的LVDS缓冲器保持了信号的完整性，并具有更低的电磁干扰和更好的电磁兼容性（EMI）及更低的电源功耗。

在数字逻辑电路的设计中，时钟Clock、复位信号往往需要作用于系统中的每个时序逻辑单元，因此在Cyclone器件中设置有全局控制信号。由于系统的时钟延时会大大影响系统的性能，在Cyclone中设置了复杂的全局时钟网络，以减少时钟信号的传输延迟，特别适合于设计同步时序逻辑电路。

Cyclone 的电源支持采用内核电压和 I/O 电压分开供电的方式，I/O 电压取决于使用时需要的 I/O 标准，而内核电压使用 1.5V 供电，Cyclone II 系列内核电压则为 1.2V，Cyclone III 系列内核电压则为 1.1V。低工作电压的应用有助于提高系统的低功耗、高速度、高集成度性能。这相对于传统的 5V 工作电压的 TTL 器件有了巨大的进步。

习　题

5-1　若主从 RS 触发器的输入波形如图 5-38 所示，试画出输出端 Q、$\overline{Q}$ 的电压波形。设触发器的初始状态 Q=0。

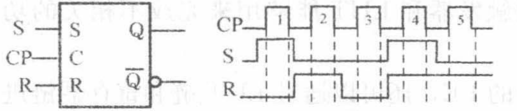


图 5-38　题 5-1 的电路与波形图

5-2　锁存器和触发器的主要区别是什么？边沿触发器与主从触发器比较，具有什么优点？

5-3　设图 5-39 中的触发器的初态均为 0，试画出 Q 端的波形。

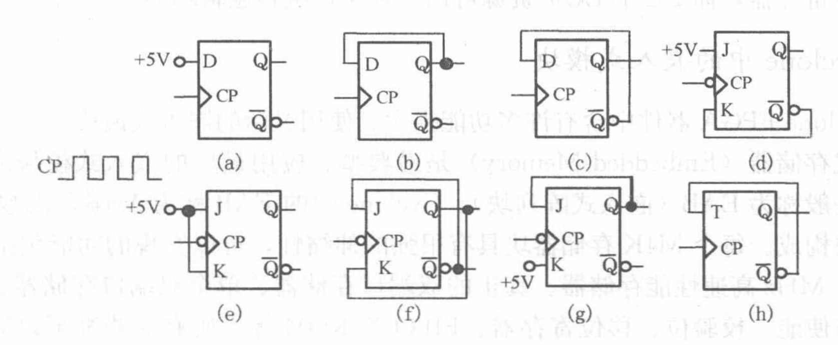


图 5-39　题 5-3 的电路与波形图

5-4　设主从 JK 触发器的原状态为 1，按照图 5-40 所给出的 J、K、CP 输入波形，画出触发器 Q 端的输出波形图。

5-5　电路图如图 5-41 (a) 所示，输入信号 CP、R 和 D 如图 5-41 (b) 所示，试画出 Q_1、Q_2 的波形图。

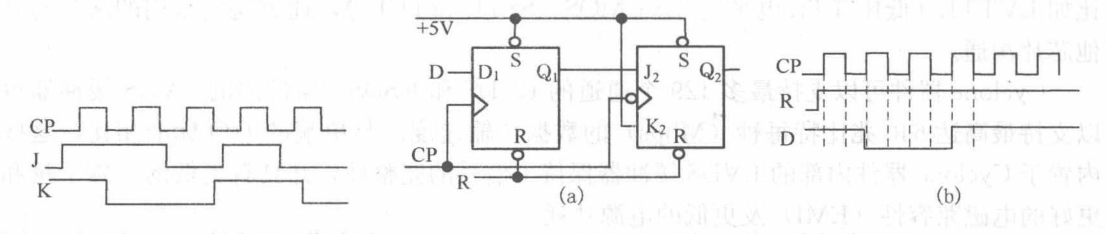


图 5-40　题 5-4 JK 触发器工作波形　　图 5-41　题 5-5 的电路与波形图

5-6　图 5-42 (a) 是一个能够产生单个脉冲的时序电路。请根据所给定的输入波形图 5-42 (b) 画出 Q_1、Q_2 的输出波形，说明电路的工作原理，并分析输出脉冲的宽度取决于什么信号。

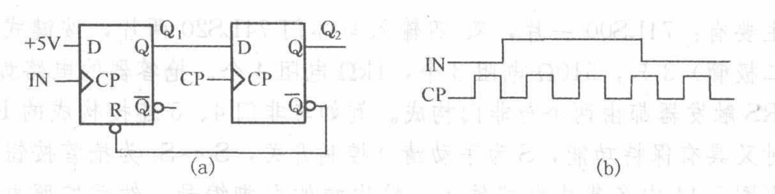


图 5-42 题 5-6 的电路与波形图

5-7 OLMC 有何功能？说明 GAL 是怎样实现可编程组合电路与时序电路功能的。

5-8 什么是基于查找表的可编程逻辑结构？

5-9 试比较 PAL、GAL、CPLD 器件及 FPGA 可编程逻辑器件各自的特点。

5-10 试用 CMOS 传输门构建边沿触发型 JK 触发器的电路。



实 验

5-1 验证集成触发器的逻辑功能及相互转换的方法

(1) 选用 TTL 器件双 D 触发器 74LS74。将 D 触发器的 D、CLK、CLRN、PRN 端分别接开关控制信号 SW1～SW4，输出端 Q 接 LED 显示。验证 D 触发器的置位、复位、同步触发功能。

(2) 选用 TTL 器件双 JK 触发器 74LS76。将 JK 触发器的 J、K、CLRN、PRN 端分别接开关控制信号 SW1～SW4，输出端 Q 接 LED 显示。验证 JK 触发器的置位、复位、保持和翻转功能。

(3) 参照图 5-29，将 JK 触发器转换成 D 触发器，并验证其功能。

(4) 参照图 5-26，将 D 触发器转换成 JK 触发器，并验证其功能。

(5) 将 D 触发器分别转换成 T 触发器和 T′触发器，并验证其功能。

(6) 将两个触发器如图 5-43 连接起来，用示波器观察记录 D 触发器的输出波形 1Q、2Q 及 CLK 波形，理解二分频和四分频的概念。

思考题：如何使用双 D 触发器构成一个三分频电路。观察和记录此电路的输入 CLK 和输出信号 1Q 和 2Q 的波形，分析记录。

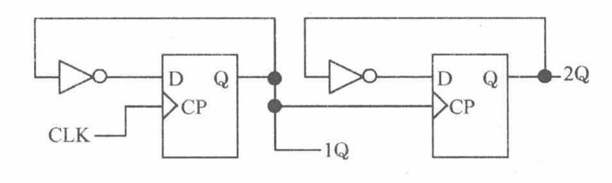


图 5-43 实验 5-1 (6) 电路

5-2 由 RS 触发器构成的多路抢答器设计

根据第 5.2.4 节的消除抖动开关的工作原理，用基本 RS 触发器设计一个三路抢答器。

使用的器件主要有：74LS00一片，双-四输入与非门74LS20两片，按键式开关4个，指示灯（发光二极管）3只，510Ω电阻3个，1kΩ电阻4个。抢答器的电路如图5-44所示，图中的每个RS触发器都由两个与非门构成。例如与非门4、5连接构成的RS触发器既有接收功能同时又具有保持功能，S为手动清0控制开关，S_1～S_3为抢答按钮开关。

首先标出图5-44中各集成电路输入、输出端的引脚编号，然后按照电路图连线，在实验系统上实现硬件验证，包括抢答功能、清0功能、互锁功能的验证。最后完成实验报告，报告中要求分析RS触发器如何实现接收、保持、输出信号功能，说明抢答器的工作原理，当抢答成功后各路信号之间是如何实现互锁功能的。

思考题1：由双输入与非门构成的保持电路，其输出状态都与哪些因素有关？试写出功能表。

思考题2：若改成六路抢答器，电路将做哪些改动？能否为抢答器增加其他功能？

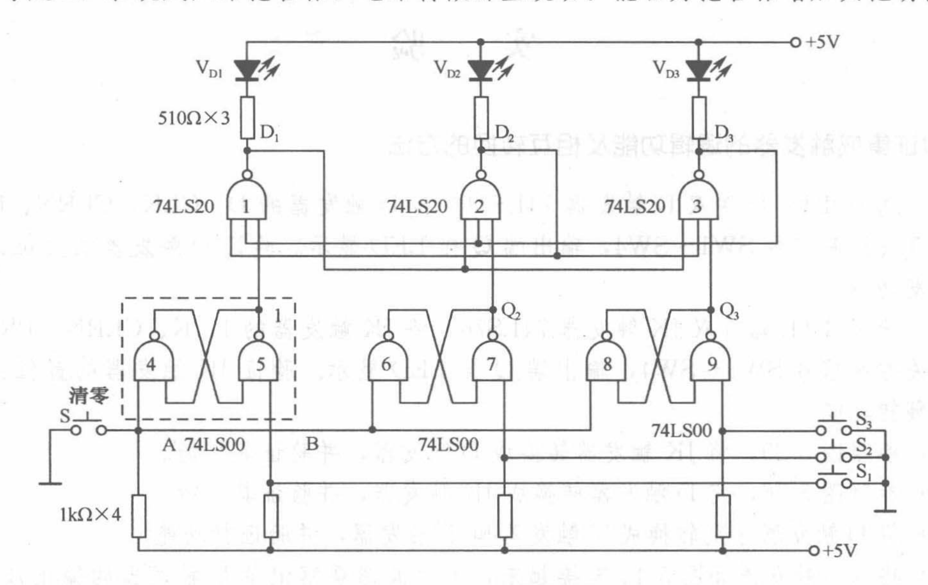


图5-44　抢答器电路图

第6章 组合电路时序分析与自动化设计

对于第 4 章中的主要内容，无论是组合电路的手工分析与设计方法，还是通用逻辑器件的使用技术或器件本身，事实上，在现代数字技术中都已很难找到它们的踪迹了。因此第 4 章的内容远非组合电路学习的全部内容，而是作为面向更新内容学习的一个台阶或桥梁，使初学者能借助于这些知识顺利地跨入现代数字技术学习和实践的领域。

本章将针对组合逻辑电路的自动时序分析和设计方法，重点介绍基于现代数字系统设计技术的自动化设计方法，其中包括相关的基础知识、数字系统自动化设计软件 Quartus II 的使用方法、组合电路的时序分析方法、硬件实现方法等。此外还要介绍基于广义译码器概念的 HDL 的逻辑真值表表述方法，从而给出了彻底、简洁而高效地解决任何类型组合电路的分析和设计方法。最后基于 Quartus II 平台，介绍几则含触发器的实用电路的设计与分析方法。

本章学习的重点是使用 Quartus II 设计组合电路以及对电路时序性能分析的方法，对于一些暂时不明了的概念或一些与设计没有直接关系的知识点可留待以后去解决，或在课余时间通过查阅有关资料去了解，因为这并不会影响当前的学习。

6.1 传统数字技术存在的问题

面对现代数字产品及其数字系统开发的要求，传统手工数字技术存在以下诸多问题而无法适应现代数字电路的开发：

(1) 低速。传统数字电路多由诸如 74 系列或 CMOS 4000 系列等通用逻辑器件构成，它们的最高工作频率仅数十兆，而现代数字系统常包含超过千兆以上的电路。

(2) 设计规模小。从卡诺图的逻辑化简应用中就能看出，卡诺图所能处理的数字电路模块的变量数非常有限，因此组合电路的设计规模只能在数十至数百个等效逻辑门规模间。然而现代数字系统的规模常超过 1000 万个逻辑门，上千个逻辑变量，这必须倚赖基于计算机的功能强大的 EDA 软件才能完成设计。

(3) 分析技术无法适应需要。首先，诸如第 4 章中给出的逻辑电路的传统分析方法只能适用于小规模逻辑电路，如果电路的规模大到数百门乃至数百万门，显然无法仅通过电路原理图来分析其逻辑功能了；其次，即使对于此类小规模逻辑电路的分析，这种传统分析方法也只能得到电路的逻辑功能，而无法获得精确的延时特性，从而无从了解电路的硬件特性、工作速度、竞争冒险情况等在实用逻辑设计中十分重要的信息。然而如果利用现代逻辑电路设计技术，对任何规模的逻辑电路，在计算机的帮助下都能高效准确地对逻辑

电路进行分析，即功能仿真与时序仿真。

(4) 效率低成本高。由于传统数字电路的设计是基于手工的，几乎整个设计过程，包括设计对象的功能表述、逻辑分析、逻辑化简、原理图绘制、电路设计，直至硬件系统的实现和测试，几乎都靠手工来完成。而且一旦在最终测试中若发现设计（包括逻辑功能和时序特性）不符合要求，还得返工，从头开始。因此即使是一个数百个逻辑门左右规模的特小型电路模块的设计周期也会很长，而设计效率的降低必然导致设计成本的提高和产品竞争力的降低。显然，相对于能满足市场要求的数百万门规模、且高效设计的现代数字产品，基于传统手工数字技术的产品没有任何生存的机会。

(5) 可靠性低。由于传统数字电路多由 74 系列或 CMOS 4000 系列等通用逻辑器件构成，规模若稍大，系统中包含的此类器件的数量将大增，这是因为系统的故障率是每一个器件故障率的总和。而现代数字产品，无论规模多大，通常仅由数片，甚至单一芯片实现，这就是所谓的片上系统（System on a Chip），因此可靠性很高。

(6) 体积大功耗大。如上所述，传统数字电路多由大量 74 系列等通用逻辑器件构成，且工作电压大多是 5V。因此体积大，功耗大，无法实现便携式产品，没有市场竞争力。现代数字系统由于能实现单片系统，且工作电压甚至可以小于 1V，因此市场前景很好。例如可以安置于人体内的心脏起搏器，不但体积微小，而且仅靠微型电池就能持续工作许多年。这是传统数字电路所无法企及的。

(7) 功能有限。由于传统逻辑器件本身功能所限，以及传统手工设计技术和检测技术落后的限制，传统数字电路的功能通常都十分简单，适用面也十分狭窄。然而基于强大的自动化数字设计技术，一个单芯片上就能实现诸如高速数字信号处理系统、功能强大的工业自动化控制系统，乃至整个计算机系统的功能。

(8) 无法实现升级。由于传统数字电路的结构和对应的功能是固定的，一旦数字电路系统设计完成，其整个功能就已确定并固定下来，如果希望改变功能，必须重新开始设计，包括整个硬件电路板的设计。然而现代数字系统中，特别是通信电路系统中，在一些实用场合，有时希望能瞬时升级硬件功能，适应新的通信协议。

(9) 知识产权不易保护。由于传统通用逻辑器件的功能是标准的，是与器件的型号一一对应的。如果都由这些器件构成系统，那么整个逻辑结构是透明的，别人很容易通过了解系统的器件组成和电路连线的关系获得整个系统的硬件组成，从而容易复制整个系统。然而现代数字系统设计者有多种方法对自己的作品进行加密。

现代数字系统自动设计技术，或电子设计自动化（EDA）技术是从传统的数字电路手工设计技术发展而来的，是现代数字电路和数字产品设计的重要工具。在数字系统设计中，它在克服了基于手工设计技术的种种缺陷后，为多功能高层次数字电子产品的设计打开了一个广阔的天地。

6.2　数字系统自动设计流程

完整地了解利用现代数字系统自动设计技术进行设计的流程，除了对于正确地使用相关的软件工具，优化设计项目，提高设计效率十分重要外，一个完整的、典型的设计流

程，同时也是 EDA 工具软件本身的组成结构。在实践中进一步了解支持这一设计流程的诸多设计工具，有利于选择好的开发工具，高效地排除设计中的问题，提高产品质量。本章给出的知识也将同步强化读者对已有的数字电路基本概念的理解和基础知识的掌握。

6.2.1 设计输入

对于目前流行的针对 PLD 的数字系统开发软件，图 6-1 的设计流程具有一般性。数字系统设计的第一步是利用输入编辑器进行图形或 HDL 文本编辑，即根据设计目标和数字系统的功能要求进行建模。即将需要实现的功能进行抽象，并用一定的方法表述出来。在此设计平台上，数字电路的功能可以表述的方法很多，例如，真值表、电路原理图、电路状态图，或者直接用特定的计算机语言表述出来。然后按照一定的要求输入计算机，以待工具软件来处理。以下对图形输入法和 HDL 文本输入法作一说明。

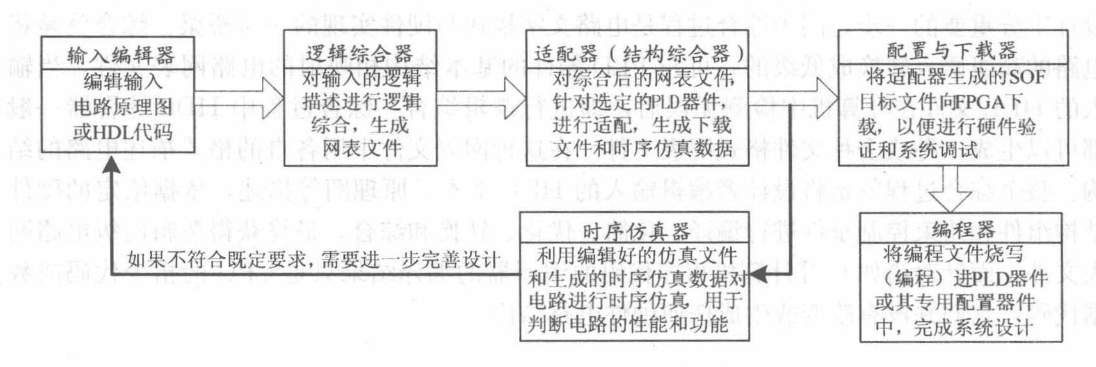


图 6-1 针对 PLD 的数字系统开发流程

1. 图形输入

图形输入通常包括原理图输入、状态图输入和波形图输入等方法。原理图输入法是一种类似于传统电子设计方法的电路原理图编辑输入方式，即在工具软件的图形编辑界面上绘制能完成特定功能的数字电路原理图。原理图由逻辑器件和连接线构成，图中的逻辑器件可以是设计软件库中预制的功能模块，如与门、非门、或门、触发器以及各种含 74 系列器件功能的宏功能块，甚至还有一些功能更强大的现成功能模块。原理图编辑绘制完成后，设计软件将对输入的图形文件进行排错，之后再将其编译成适用于逻辑综合的网表文件。

用原理图表达的输入方法的优点是不需要增加新的相关知识，设计过程形象直观，适用于初学者；缺点是不容易修改，不具有通用性，即在一种设计软件上编辑的电路通常无法被另一种设计软件识别和处理，而且只适用于较小规模数字系统的设计。

2. HDL 文本输入

这种方式与计算机软件语言编辑输入基本一致，就是将使用了某种硬件描述语言（HDL）的电路设计文本，如 VHDL 或 Verilog HDL 的源程序，进行编辑输入。应用 HDL 的文本输入方法克服了原理图输入法存在的所有弊端，成为专业设计最常用的方法。

6.2.2　硬件描述语言

在前面多次提到硬件描述语言 HDL。目前，硬件描述语言 VHDL 和 Verilog HDL 在现在数字系统自动化设计中使用得最多，也拥有几乎所有的主流设计工具软件的支持。为了便于相关的说明和读者的深入学习，本书中将会在极浅的层次上适当涉及 Verilog 的内容，但绝不深究，因此不需要准备相关的基础知识。

6.2.3　综合

图 6-1 所示流程的第 2 步是综合，即逻辑综合，完成综合的软件称综合器。

一般来说，综合是仅对应于 HDL 而言的。利用 HDL 综合器对设计进行综合是自动设计十分重要的一步，因为综合过程是电路文字描述与硬件实现的一座桥梁。综合就是将电路的高级语言转换成低级的，可与 PLD 器件的基本结构相映射的电路网表文件。当输入的 HDL 文件在计算机中检测无误后，即进行逻辑综合。综合过程中 HDL 综合器一般都可以生成一种或多种文件格式网表文件，在这种网表文件中用各自的格式描述电路的结构。整个综合过程就是将设计者编辑输入的 HDL 文本、原理图等描述，依据给定的硬件结构组件和约束控制条件进行编译、化简、优化、转换和综合，最终获得逻辑门级电路网表文件。而针对诸如 C 等计算机程序代码的编译器的编译结果只是 CPU 的指令代码或数据代码，它们并没有改变或生成任何硬件电路结构。

6.2.4　适配

图 6-1 所示流程的第 3 步是适配，完成适配的软件称适配器。适配器一般也称逻辑结构综合器，它的功能是将由综合器产生的描述电路连接关系的网表文件配置（安排放置的意思）于指定的目标器件中，如 PLD 器件中，使之产生最终的下载文件。适配器将综合后网表文件针对某一具体的目标器件（当前设计选中的 PLD 器件）进行逻辑映射操作，其中包括底层器件配置、逻辑分割、逻辑优化、逻辑布局布线操作。适配完成后可以利用适配所产生的仿真文件作精确的时序仿真，同时产生可用于编程下载的文件。

6.2.5　仿真

图 6-1 所示流程的第 4 步是仿真，完成仿真的软件称仿真器。在最后的硬件系统实现，即在编程下载和硬件系统测试前必须利用对适配生成的结果进行模拟条件下的逻辑电路的功能和性能进行测试与评估（即仿真）。仿真就是让计算机根据一定的算法和一定的仿真库对系统设计进行模拟，以验证设计的正确性。仿真是在自动化设计过程中的重要步骤。

仿真软件含有时序仿真功能，时序仿真能完成接近真实器件运行特性的仿真。由于仿真文件中已包含了目标器件的硬件特性参数，因而仿真精度很高，即在计算机上的仿真的

结果可以与后期对应的硬件系统的实测结果有好的吻合。

时序仿真是自动设计技术最优秀的特性和最重要的硬件调试工具之一。传统的手工设计技术最缺乏的正是针对所设计的数字系统的硬件延迟特性的测试和表达手段，从而导致了一系列的设计缺陷和技术缺陷，因此当计算机一经普及，传统的手工数字技术即被淘汰了。

6.2.6 硬件测试

图 6-1 所示流程的第 5 步是编程下载，完成编程下载的软件称编程器。

把适配后生成的下载或配置文件，通过编程器或编程电缆向 FPGA 或 CPLD 等 PLD 器件进行下载，以便进行硬件调试和验证（Hardware Debugging）。

最后是将含有整个设计系统的 FPGA 或 CPLD 的硬件系统进行统一测试，以便最终在硬件环境中验证设计项目实际工作情况，以便排除错误，完善设计。

6.3 原理图输入法逻辑电路设计

本节拟使用原理图输入的设计方式，通过完成一个简单组合电路的设计和测试，详细介绍 Quartus II 的完整设计流程，使读者能够迅速掌握利用 Quartus II 完成数字系统自动化设计的基本方法，并利用这些方法更有效和更深入地理解和掌握之前遇到的数字电路的基础知识和实用设计技术。

其实以下介绍的设计流程具有一般性，它同样适用于其他输入方法的设计，如基于 HDL 的硬件描述语言的输入或混合输入等设计方法。

6.3.1 原理图编辑输入方法

本节的示例是设计一个简单的脉冲发生器，电路如图 6-2 所示。其中 A、B、C 是输入信号，Y[7..0] 是输出信号，P 是脉冲输出端。其中 Y[7..0] 是总线表述方法，它是 8 个单独信号 Y[7]、Y[6]、…、Y[0] 的集合表示方法。本示例的目标是完成对图 6-2 电路的编辑输入、时序仿真、硬件实现和功能测试。

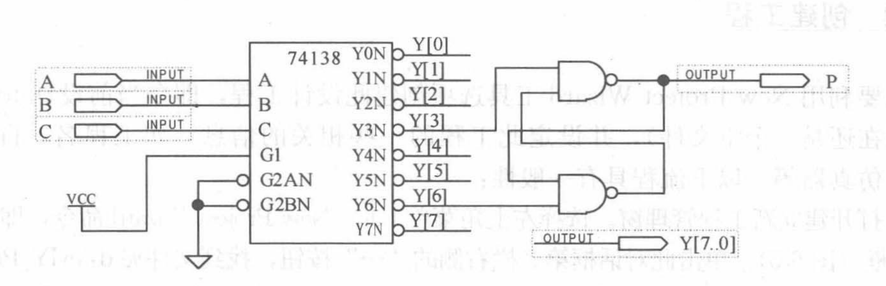


图 6-2 脉冲发生器示例电路图

在将此电路原理图在计算机上编辑输入前，首先需要建立工作库文件夹，以便存储工

程项目设计文件。任何一项设计都是一项工程（Project），都必须首先为此工程建立一个放置与此工程相关的所有设计文件的文件夹。此文件夹将被 Quartus II 默认为工作库（Work Library）。一般，不同的设计项目最好放在不同的文件夹中，而同一工程的所有文件都必须放在同一文件夹中。设计中不要将文件夹设在计算机已有的安装目录中，更不要将工程文件直接放在根目录中，此外，文件夹名不能用中文，也最好不要用数字。

在建立了文件夹后就可以通过 Quartus II 的原理图编辑器编辑电路了，步骤如下：

（1）新建一个文件夹。首先可以利用 Windows 资源管理器来新建一个文件夹。这里假设本项设计的文件夹取名为 MY_PROJECT，在 D 盘中，路径为 d:\MY_PROJECT。

（2）打开原理图编辑窗。打开 Quartus II，选择左上角菜单 File→New。在 New 窗口中的 Design Files 条目中选择原理图文件类型，这里选择 Block Diagram/Schematic File（图 6-3）。然后即可在此原理图编辑窗（图 6-4）中加入所需的电路元件。

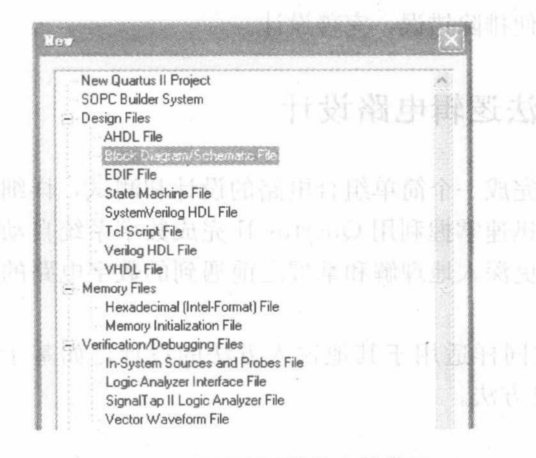


图 6-3 选择原理图编辑文件类型

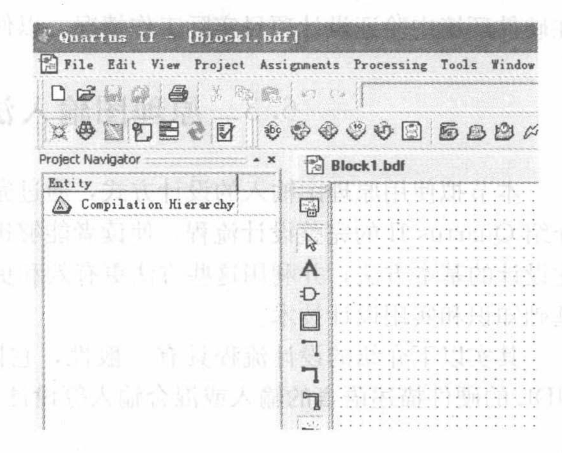


图 6-4 打开原理图编辑窗

（3）文件存盘。选择 File→Save As 命令，找到已设立的文件夹路径为：d:\MY_PROJECT，将此空原理图文件存盘。存盘文件名可取为 top. bdf。若出现问句“Do you want to create…”时，若单击“是”按钮，则直接进入创建工程流程；若单击“否”按钮，可按以下的方法进入创建工程流程。本示例中选择“否”。

6.3.2 创建工程

在此要利用 New Project Wizard 工具选项创建此设计工程，即令当前设计 top. bdf 为工程（现在还是一个空文件），并设定此工程的一些相关的信息，如工程名、目标器件、综合器、仿真器等。以下流程具有一般性：

（1）打开建立新工程管理窗。选择左上角菜单 File→New Project Wizard 命令，即弹出工程设置对话框（图 6-5）。单击此对话框第 2 栏右侧的“…”按钮，找到文件夹 d:\MY_PROJECT，选中已存盘的文件 top. bdf，再单击“打开”按钮，即出现如图 6-5 所示的设置情况。其中第 1 栏的 d:\MY_PROJECT 表示工程所在的工作库文件夹；第 2 栏的 top 表示此项工程的工程名，工程名可以取任何其他的名，也可直接用顶层文件名或实体名（如果是

HDL 文件的话）；第 3 栏是当前工程顶层文件的实体名，这里即为 top。

图 6-5 利用 New Project Wizard 创建工程 top

（2）将设计文件加入工程中。单击下方的 Next 按钮，在弹出的对话框中单击 File name 栏的按钮“…”，将与工程相关的文件（如 top.bdf）加入进此工程，按右侧的“Add…”按钮。

（3）选择目标芯片。单击 Next 按钮，选择目标芯片。首先在 Family 栏选芯片系列，在此选 Cyclone III 系列，准备选择的目标器件是 EP3C10E144C8（也可选择其他系列 FPGA，这里假设实验台上含有此型号的 FPGA）。这里的 EP3C10 表示 Cyclone III 系列及此器件的逻辑规模；E 表示带有金属地线底板的 TQFP 封装；C8 表示速度级别。

（4）工具设置。单击 Next 按钮后，弹出的下一个窗是软件工具设置窗：EDA Tool Settings 窗。在此选择默认，表示仅选择 Quartus II 自含的所有设计工具。

（5）结束设置。单击 Next 按钮后即弹出“工程设置统计”窗口，上面列出了此项工程相关设置情况。最后单击 Finish 按钮，即已设定好此工程，并出现 top 的工程管理窗，或称 Compilation Hierarchies 窗口，主要显示本工程项目的层次结构（图 6-6）。注意此工程管理窗左上角所示的工程路径、工程名 top、右下窗眉处是已打开的文件名。

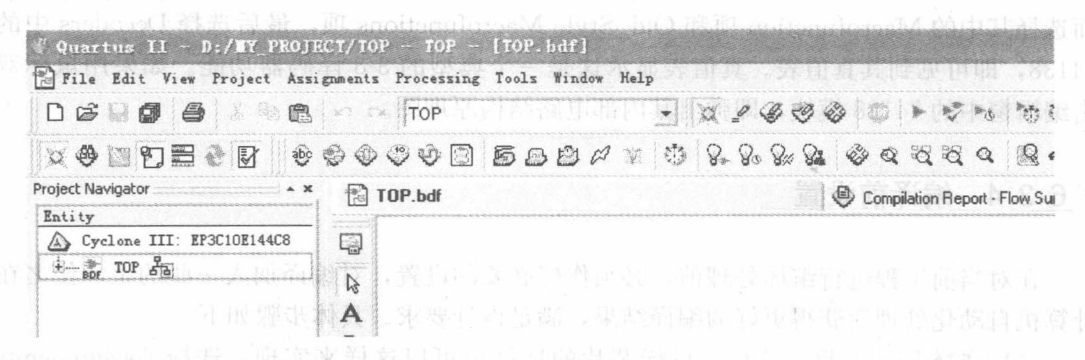


图 6-6 top 工程管理窗

（6）编辑构建电路图。如图 6-6 所示，双击左栏的工程名称 TOP，于是打开原理图编辑窗；再于此编辑窗内任何一点双击，将弹出一个逻辑电路器件输入对话框（图 6-7）。在此对话框的左栏 Name 框内键入所需元件的名称，在此为 74138。

由于仅考虑器件的逻辑功能，同类功能的器件，如 74LS138、74HC138、74S138 等，

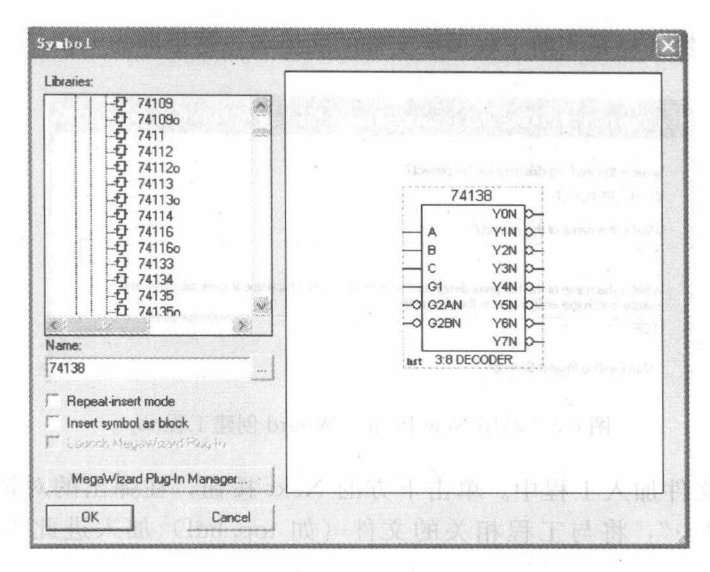


图 6-7 在元件调用对话框调出宏功能元件 74138

一律命名为 74138。然后单击 OK 按钮，即可将此元件调入编辑窗中。再以同样方法调入两个 2 输入与门，名称是 AND2，以及输入输出端口，名称分别是 INPUT 和 OUTPUT。输入输出端口的名称可以通过双击端口元件，在弹出的对话框中键入，如 A。最后根据图 6-2 直接用鼠标拖出连线将它们连接起来。

6.3.3 功能分析

图 6-2 所示电路的主要器件是 74138，这里首先了解元件 74138 的功能。为此打开帮助文件 Macrofunctions（查 www. kx-soc. com 可获此文件），打开后选择 Messages 项，继而选择其中的 Macrofunction 项和 Old_Style Macrofunctions 项，最后选择 Decoders 中的 74138，即可见到其真值表。真值表显示这是一个典型的 3-8 译码器功能。如果用鼠标双击编辑窗中的 74138 模块，即弹出其内部电路结构原理图。

6.3.4 编译前设置

在对当前工程进行编译处理前，必须作好必要的设置，对编译加入一些约束，使之在计算机自动化处理后获得更好的编译结果，满足设计要求。具体步骤如下：

（1）选择 FPGA 目标芯片。目标芯片的选择也可以这样来实现：选择 Assignmemts 菜单中的 Settings 项，在弹出的对话框中（图 6-8）选择 Category 项下的 Device。选择需要的 FPGA 目标芯片，如 EP3C10E144C8（此芯片已在建立工程时选定了）。

（2）选择配置器件的工作方式。单击图 6-8 中的 Device and Pin Options 按钮，进入 Device and Pin Options 选择窗（图 6-9）。在此首先选择 General 项。可以在此做一些必要的选择，具体情况可以注意窗口下方显示的帮助说明。

(3) 双功能输入输出端口设置。选择双目标端口设置页 Dual-Purpose Pins（图 6-10），将 nCE0 原来的 Use as programming pin 改为 Use as regular I/O（nCE0 端口作编程口时，可用于多 FPGA 芯片的配置）。这样可以将此端口也作普通 I/O 口来用。请特别关注此项设置必须事先完成，否则一旦在编译时报错，软件给出的报告不会明示错误所在，初学者很难判断问题原因。对于已选的 EP3C10 器件，此双功能脚是 Pin 101，需关注此引脚的编译情况（如果用到此引脚的话）。

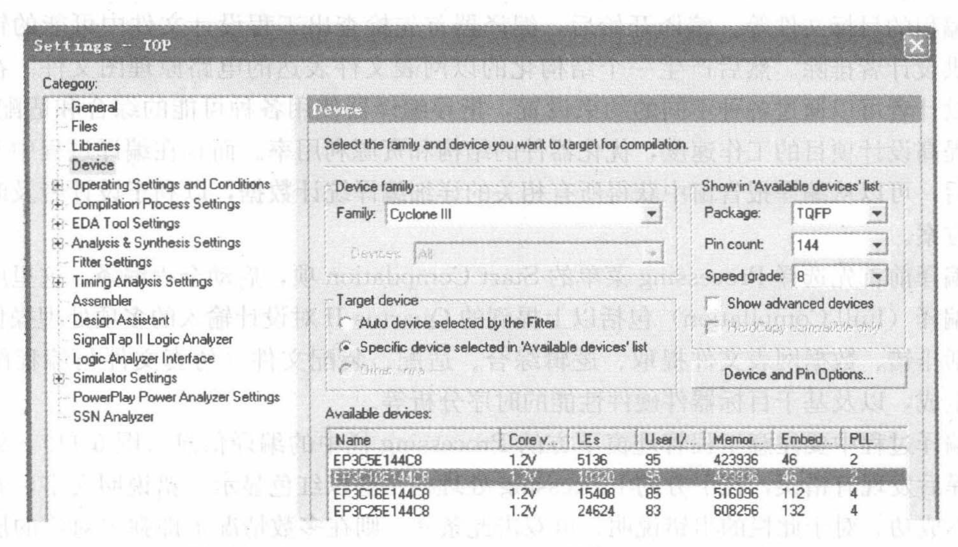


图 6-8 由 Settings 对话框选择目标器件 EP3C10E144C8

图 6-9 选择配置器件型号和压缩方式

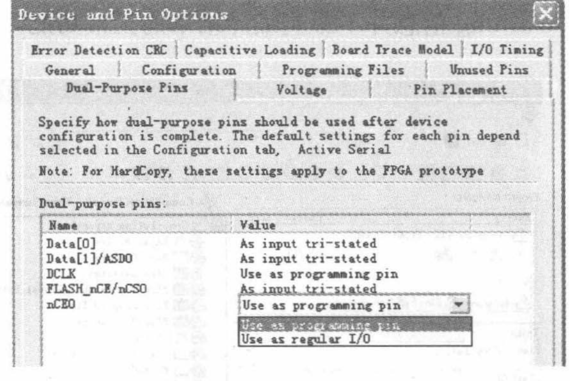


图 6-10 指定 nCE0 脚为普通 I/O 端口

(4) 选择目标器件闲置引脚的状态。此项选择在某些情况下十分重要。选择图 6-10 所示窗口的 Unused Pins 项，此页中可根据实际需要选择目标器件闲置引脚的状态，可选择为输入状态（呈高阻态，推荐此项选择），或输出状态（呈低电平），或输出不定状态；或不作任何选择。

在其他页也可作一些选择，各选项的功能可参考窗口下方的 Description 说明。

6.3.5 全程编译

Quartus II 编译器是由一系列处理模块构成的，这些模块负责对设计项目的检错，逻辑综合、结构综合、输出结果的编辑配置，以及时序分析等。在这一过程中，为了把设计项目适配到 FPGA 目标器中，将同时产生不同用途的输出文件，如功能和时序信息文件、器件编程的目标文件等。编译开始后，编译器首先检查出工程设计文件中可能的错误信息，供设计者排除。然后产生一个结构化的以网表文件表达的电路原理图文件。在编译前，设计者可以通过各种不同的约束设置，指导编译器使用各种可能的综合和适配技术，以便提高设计项目的工作速度，优化器件的结构和资源利用率。而且在编译过程中及编译完成后，可以从编译报告窗中获得所有相关的详细编译统计数据，以利于设计者及时调整设计方案。

编译前首先选择 Processing 菜单的 Start Compilation 项，启动全程编译。这里所谓的全程编译（Full Compilation）包括以上提到的 Quartus II 对设计输入的多项处理操作，其中包括排错、数据网表文件提取、逻辑综合、适配、装配文件（仿真文件与编程配置文件）生成，以及基于目标器件硬件性能的时序分析等。

编译过程中要注意工程管理窗下方的 Processing 栏中的编译信息（图 6-11）。如果启动编译后发现有错误，在下方的 Processing 处理栏中会以红色显示出错说明文字，并告知编译不成功。对于此栏的出错说明，可双击此条文，则在多数情况下即弹出对应的层次的文件，并用深色标记指出错误所在。改错后再进行编译直至排除所有错误。

Processing 栏出现的 Warning 报警信息以蓝色文字出现，但也要充分注意。有的 Warning 信息并不影响编译结果的正常功能，但有的则不然。

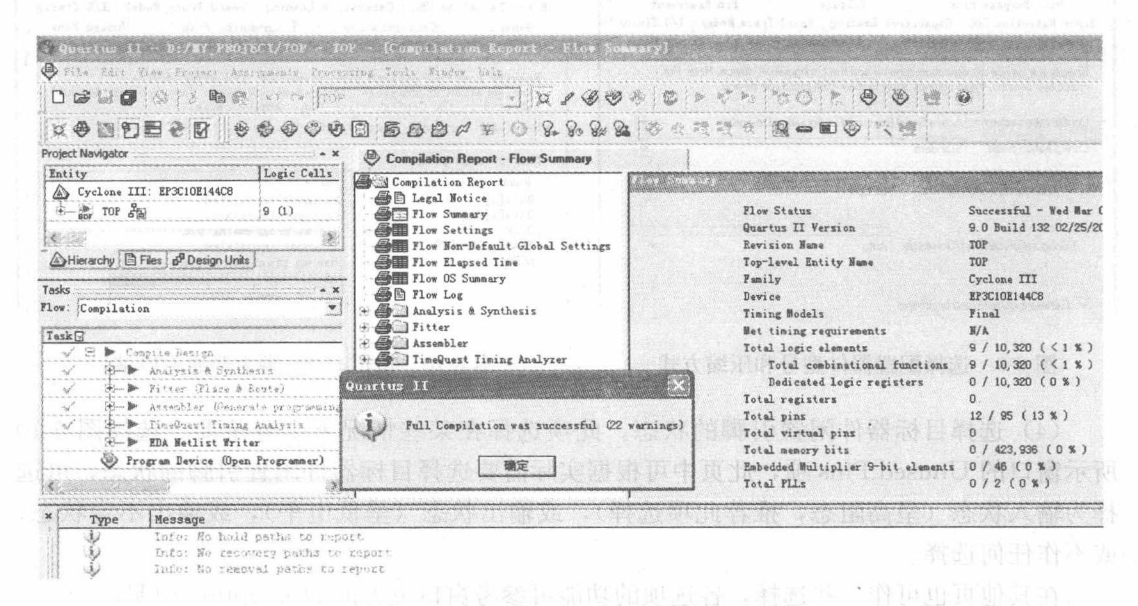


图 6-11 全程编译后工程管理窗的情况

如果编译成功，可以见到如图 6-11 所示的工程管理窗的左上角显示出工程 top 的层次结构和其中结构模块耗用的逻辑宏单元数（图 6-11 显示 9 个）；在此栏下是编译处理流程，包括数据网表建立、逻辑综合、适配、配置文件装配和时序分析等。最下栏是编译处理信息；中栏，即 Compilation Report 栏，是编译报告项目选择菜单，单击其中各项可以详细了解编译与分析结果。

例如单击 Flow Summary 项，将在右栏显示硬件耗用统计报告，其中报告了当前工程耗用了 9 个逻辑宏单元、0 个 D 触发器、0 个内部 RAM 位等。

6.3.6 逻辑功能测试

工程编译通过后，必须对其功能和时序性质进行测试，以了解设计结果是否满足原设计要求。对以上工程项目的仿真测试流程的详细步骤如下：

（1）打开波形编辑器。选择菜单 File 中的 New 项，在 New 窗口中选择 Verification 项下的 Vector Waveform File（图 6-3），单击 OK 按钮，即出现空白的仿真波形编辑器，注意将窗口扩大，以利观察。

（2）设置仿真时间区域。对于时序仿真来说，将仿真时间轴设置在一个合理的时间区域上十分重要。通常设置的时间范围在数十微秒间：在 Edit 菜单中选择 End Time 项，在弹出的窗口中的 Time 栏处输入 75，单位选 us（在仿真软件中用 u 代表 μ），整个仿真域的时间即设定为 75μs（图 6-12），单击 OK 按钮，结束设置。

（3）波形文件存盘。选择 File 中的 Save as，将以默认名为 top.vwf 的波形文件存入文件夹 d:\MY_PROJECT 中。

（4）将工程 top 的端口信号名选入波形编辑器中。方法是首先选择 View →Utility Windows→Node Finder 选项。弹出的对话框如图 6-13 所示，在 Filter 框中选“Pins：all”（通常已默认选此项），然后单击 List 按钮，于是在下方的 Nodes Found 窗口中出现设计中的 top 工程的所有端口名。通常希望 Node Finder 窗是浮动的，可以用右键单击此窗边框，在弹出的小窗上消去 Enable Docking 选项即可。注意如果此对话框中的 List 不显示 top 工程的端口引脚名，需要重新编译一次，即选择 Processing→Start Compilation，然后再重复以上操作过程。

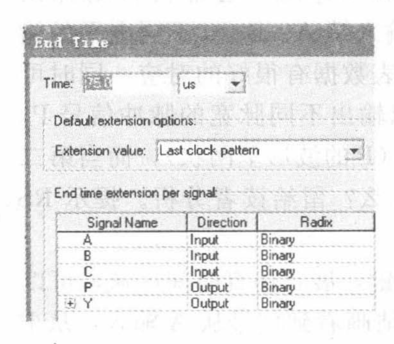


图 6-12 设置仿真时间长度

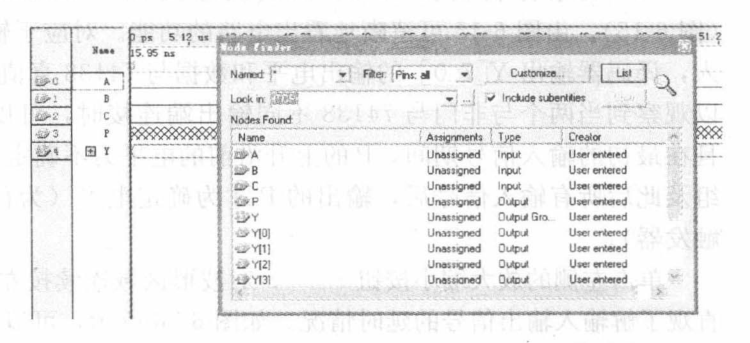


图 6-13 从 Node Finder 窗向波形编辑器拖入信号节点

最后，用鼠标将重要的端口名A、B、C、P和输出总线信号Y分别拖到波形编辑窗，结束后关闭Nodes Found窗口。再单击波形窗左侧的全屏显示按钮，使全屏显示，并单击放大缩小按钮后，再用鼠标在波形编辑区域右键单击，使仿真坐标处于适当位置，如图6-13上方所示，这时仿真时间横坐标设定在数十微秒数量级。

（5）编辑输入波形（输入激励信号）。即对输入引脚A、B、C设置输入信号，以便通过观察输出信号来了解电路的功能。如图6-14所示，用鼠标拖的方式，使之变成蓝色条，再单击左列的高电平设置按钮“1”，最后如图6-14所示。为了容易输入数据和辨认，可以将C、B、A这3个信号合并（Grouping）起来。方法是将这3个信号拉成块，再右键单击，在出现的菜单中选择Grouping，并取名，如CBA。对于输入信号CBA置数时，可以先选中CBA，再通过单击工具栏的“C”来置数。

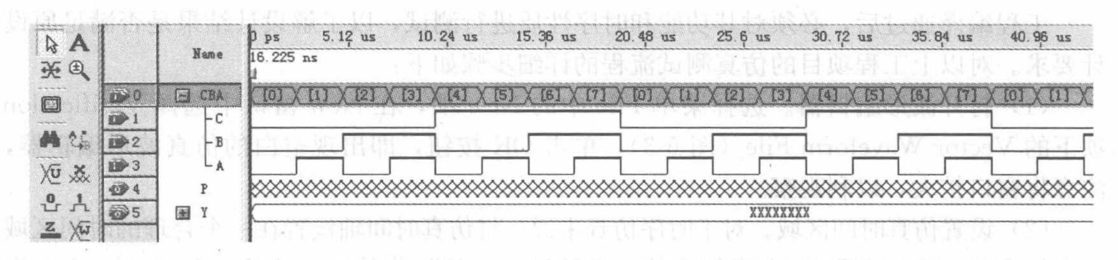


图6-14　设置好的仿真文件激励波形图

单击如图6-14所示的输出信号Y左旁的“+”号，则能展开此总线中的所有信号；如果双击此“+”号左旁的信号标记“O 5”，将弹出对该信号数据格式设置的对话框。在该对话框的Radix栏有7种选择，这里选择默认的二进制数Binary表达方式。最后设置好的激励信号波形图如图6-14右侧所示。完成后须对波形文件再次存盘。

（6）仿真器参数设置。选择菜单Assignments中的Settings，在Settings窗口下选择Category→Simulator Settings，在右侧的Simulation mode项下选择Timing（通常默认），即选择时序仿真，并选择仿真激励文件名top.vwf（通常默认）；选中Run simulation until all vector stimuli are used全程仿真等。

（7）启动仿真器。现在所有设置进行完毕。在菜单Processing项下选择Start Simulation，直到出现Simulation was successful，仿真结束。

（8）观察分析仿真结果。仿真波形文件“Simulation Report”通常会自动弹出（图6-15）。由图6-15可清晰地看出电路的功能。对应于输入端A、B、C不同电平的输入，译码器输出Y[7..0]的输出电平和数据与74138真值表数据有很好的对应。同时可以观察到当两个与非门与74138不同输出端连接时，可以输出不同脉宽的脉冲信号P，且在最初的输入信号期间，P的上升沿前的电平为不确定（P的波形呈网状），而当第二组及此后所有输入信号后，输出的P才为确定电平（为什么？留给读者分析。提示RS触发器）。

单击左侧的放大缩小按钮+/－，对波形区域连续按左键，展开仿真时间区域，可以直观了解输入输出信号的延时情况。如图6-16所示，可以清晰看到信号从A输入，从P口输出的延时情况。为了精确测量，先单击左侧的箭头按钮，再对图6-16的A输入上升

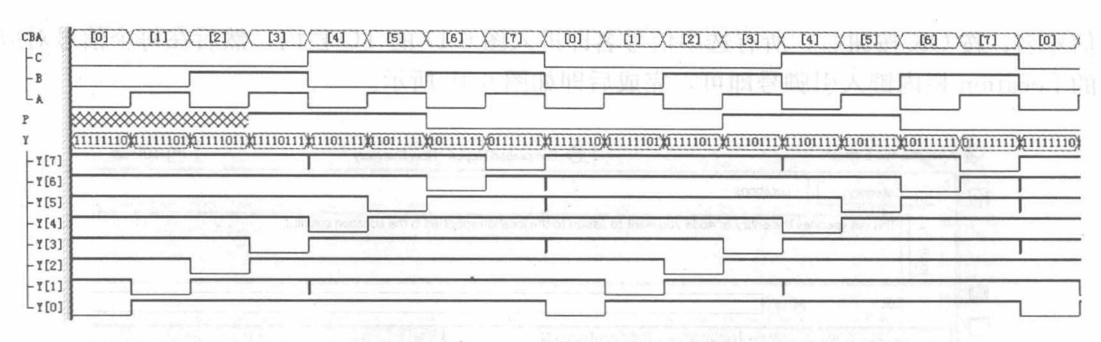


图 6-15 图 6-2 电路的仿真波形输出

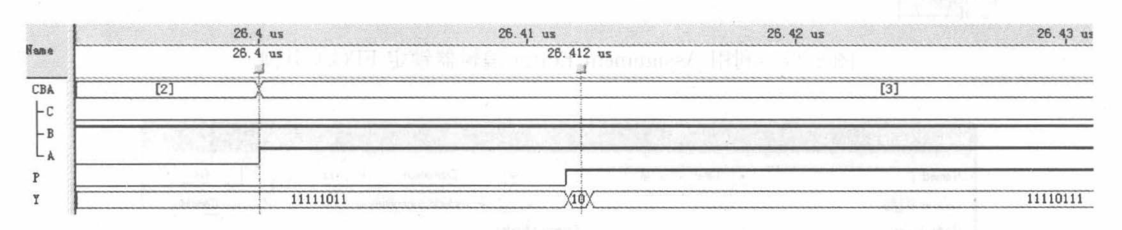


图 6-16 输入 A 与输出 P 的延时波形显示

沿处右键单击，然后对弹出的菜单选择 Insert Time Bar，即出现如图中所示的标有 26.4us 的时间指示，对于 P 的上升沿也如此。两者之差即为此电路当实现于 EP3C10E144C8 器件中两个指定端口的精确延时量，这正是时序仿真的优势。

6.4 硬件测试

为了能对此逻辑模块进行硬件测试，应将其输入输出信号锁定在 FPGA 芯片确定的引脚上，编译后下载。主要步骤如下。

6.4.1 引脚锁定

为了便于说明，在此借助 KX-7C10E＋系统（参考附录），以下简称为 10E＋系统。如果读者有其他 FPGA 开发板，也一样可以用来验证，但须注意不同的引脚锁定。

需要锁定的引脚分别为：输入信号引脚 A、B、C，分别锁定于拨码开关的 3 个控制端：P89、P90、P91；译码输出 Y[7]、Y[6]、…、Y[0] 分别锁定于开发板上的 8 个发光管：P11、P10、P7、P4、P3、P2、P1、P144；脉冲输出 P 锁定于数码管 LEDC（最下方的数码管）的小数点：P49。确定了锁定引脚编号后就可以完成以下引脚锁定操作了：

（1）打开工程。在菜单 File 中选择 Open Project 项，并选择工程文件 top，打开此前已设计好的工程。选择 Assignments 菜单中的 Assignment Editor 项，即进入如图 6-17 所示的 Assignment Editor 编辑窗。在 Category 栏中选择 Locations。

（2）双击 TO 栏的《new》，即出现一按钮，单击此按钮，并选择出现的菜单中的 Node Finder 项（图 6-17）。在弹出的如图 6-18 所示的对话框中选择本工程要锁定的端口

信号名，按 OK 按钮后，所有选中信号名即进入图 6-17 的 TO 栏内。然后在每个信号对应的 Location 栏内键入引脚号即可，完成后即如图 6-19 所示。

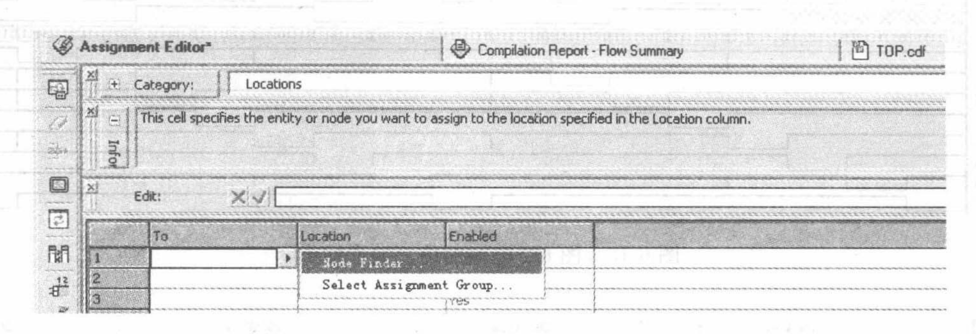


图 6-17　利用 Assignment Editor 编辑器锁定 FPGA 引脚

Node Finder

Named: *　Filter: Pins: all　Customize...　List　OK
Look in: |TOP|　Include subentities　Stop　Cancel

Nodes Found:

Name	Assignments	T
A	PIN_89	Ir
B	PIN_90	Ir
C	PIN_91	Ir
P	PIN_49	C
Y	Unassigned	C
Y[0]	PIN_144	C
Y[1]	PIN_1	C
Y[2]	PIN_2	C
Y[3]	PIN_3	C
Y[4]	PIN_4	C
Y[5]	PIN_7	C
Y[6]	PIN_10	C
Y[7]	PIN_11	

Selected Nodes:

Name	Assignments	T
|TOP|A	PIN_89	Ir
|TOP|B	PIN_90	Ir
|TOP|C	PIN_91	Ir
|TOP|P	PIN_49	O
|TOP|Y[0]	PIN_144	O
|TOP|Y[1]	PIN_1	O
|TOP|Y[2]	PIN_2	O
|TOP|Y[3]	PIN_3	O
|TOP|Y[4]	PIN_4	O
|TOP|Y[5]	PIN_7	O
|TOP|Y[6]	PIN_10	O
|TOP|Y[7]	PIN_11	O

图 6-18　选择需要锁定的引脚信号

	To	Location
1	A	PIN_89
2	B	PIN_90
3	C	PIN_91
4	P	PIN_49
5	Y[0]	PIN_144
6	Y[1]	PIN_1
7	Y[2]	PIN_2
8	Y[3]	PIN_3
9	Y[4]	PIN_4
10	Y[5]	PIN_7
11	Y[6]	PIN_10
12	Y[7]	PIN_11
13	<<new>>	

图 6-19　引脚锁定对话框

(3) 最后保存这些引脚锁定的信息后，必须再编译(启动 Start Compilation) 一次，才能将引脚锁定信息编译进编程下载文件中。此后就可以准备将编译好的 SOF 文件下载到实验系统的 FPGA 中了。

6.4.2　对 FPGA 编程配置

引脚锁定并编译完成后，Quartus II 将生成多种形式的针对所选目标 FPGA 的编程文件。其中最主要的是 POF 和 SOF 文件，前者是编程目标文件，用于对配置器件编程；后者是静态 SRAM 目标文件，用于对 FPGA 直接配置，在系统直接测试中使用。这里首先将 SOF 格式配置文件通过 JTAG 口载入 FPGA 中进行硬件测试。步骤如下：

(1) 打开编程窗。首先将实验系统和 USB-Blaster 编程器连接，打开电源。在菜单 Tools 中选择 Programmer，弹出如图 6-20 所示的编程窗。在 Mode 栏选 JTAG（默认），

并选中打勾下载文件右侧的第一小方框。注意要核对下载文件路径与文件名。如果此文件没有出现或有错，单击左侧 Add File 按钮，手动选择配置文件 top.sof。

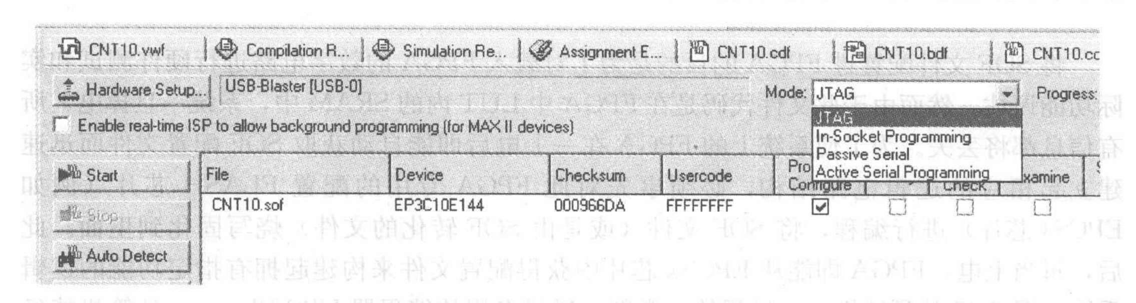


图 6-20　选择编程下载文件和下载模式

(2) 设置编程器。若是初次安装的 Quartus II，在编程前必须进行编程器选择操作。若准备选择 USB-Blaster 编程器。单击 Hardware Setup 按钮（图 6-20）可设置下载接口方式，在弹出的 Hardware Setup 对话框中（图 6-21），选择 Hardware Settings 页，再双击此页中的选项 USB-Blaster 之后，单击 Close 按钮，关闭对话框即可。这时应该在编程窗右上显示出编程方式：USB-Blaster。

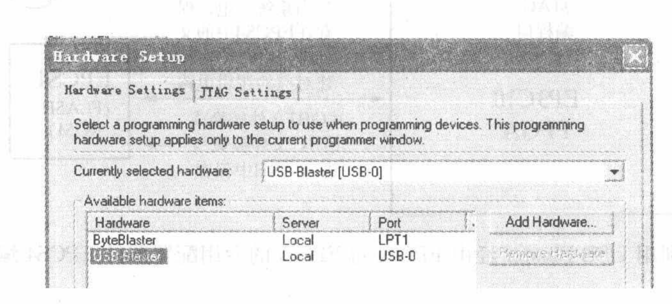


图 6-21　加入编程下载方式

最后单击下载标符 Start 按钮，即进入对目标器件 FPGA 的配置下载操作。当 Progress 显示出 100%，或处理信息栏中出现“Configuration Succeeded”时，表示编程成功。

(3) 测试 JTAG 口。如果要测试 USB-Blaster 编程器与 FPGA 的 JTAG 口是否连接好，可以单击图 6-20 所示编程对话框的 Auto Detect 按钮，看是否能读出实验系统上 FPGA 的型号。

(4) 硬件测试。成功下载 top.sof 后，拨动拨码开关对应开关，观察 8 个发光管和 P 对应的数码管小数点的亮灭情况，判断 FPGA 中模块的逻辑功能和工作情况。

注意：*在初次使用 USB-Blaster 编程器前，需首先安装驱动程序：将 USB-Blaster 编程器一端插入 USB 口，这时会弹出一个 USB 驱动程序对话框，根据对话框的引导，选择用户自己搜索驱动程序，这里假定 Quartus II 安装在 E 盘，则驱动程序的路径为 E:\altera\quartus90\drivers\usb-blaster。安装完毕后，打开 Quartus II 选择编程器，单击左上角的 Hardware Setup 按钮，在弹出的窗口中选择 USB-Blaster 项，双击之。此后就能按照前面介绍的方法使用了。*

6.4.3 对FPGA配置器件编程

将SOF文件配置进FPGA的目的是为了对载入FPGA的数字电路进行硬件测试和实际功能评估，然而由于此文件代码是在FPGA中LUT内的SRAM中，系统一旦掉电，所有信息都将丢失。为了使系统上的FPGA在一上电后即能自动获取SOF配置文件而迅速建立起相应的逻辑电路结构，必须事先对此FPGA专用的配置FLASH芯片（例如EPCS4芯片）进行编程，将SOF文件（或是由SOF转化的文件）烧写固化到里面。此后，每当上电，FPGA即能从EPCSx芯片中获得配置文件来构建起拥有指定功能的逻辑系统。图6-22就是这样一个过程的示意图。通过专用的编程器USB-Blaster，计算机能经由JTAG口，与FPGA建立起双向联系。通过这条通道，计算机可以向FPGA直接配置SOF文件，也能向EPCS器件烧写文件，或实时收集内部系统的运行信息，还能对FPGA的内部逻辑及嵌入的各种功能模块进行测控。

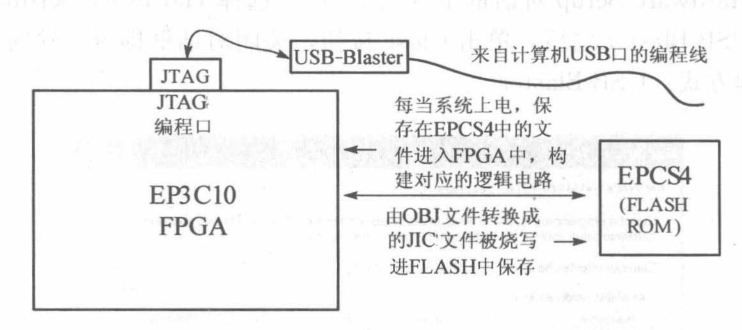


图6-22 利用USB-Blaster经由FPGA向FPGA的专用配置器件EPCS4编程下载

以下介绍利用JTAG口对EPCS器件进行编程的方法。具体流程是首先将SOF文件转化为JTAG间接配置文件，再通过FPGA的JTAG口为EPCS器件编程，步骤如下：

（1）将SOF文件转化为JTAG间接配置文件。选择File→Convert Programming Files命令，在弹出的窗口（图6-23）中作如下选择：

① 首先在Programming file type下拉列表框中选择输出文件类型为JTAG间接配置

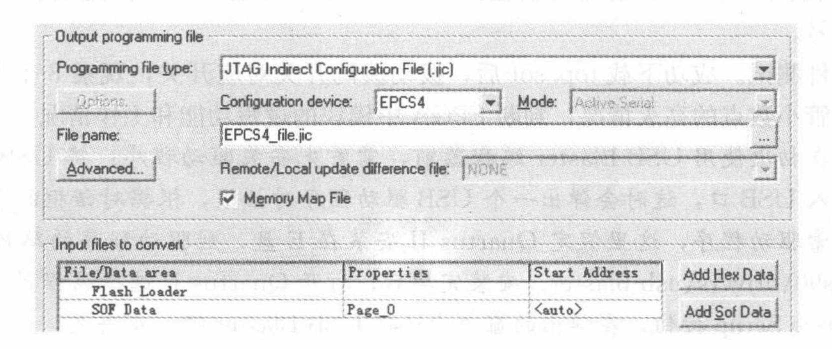


图6-23 设定JTAG间接编程文件

文件类型：JTAG Indirect Configuration File，后缀：. jic。

② 然后在 Configuration device 下拉列表中选择配置器件型号，这里选择 EPCS4。

③ 再于 File name 文本框中键入输出文件名，如 EPCS4_file. jic。

④ 选择下方 Input files to convert 栏中的 Flash Loader，再选择右侧的 Add Device 按钮，对于弹出的 Select Devices 器件选择窗（图 6-24），于左栏中选定目标器件的系列，如 Cyclone III；再于右栏中选择具体器件 EP3C10。选中 Input files to convert 栏中的 SOF Data 项，然后单击右侧的 Add File 按钮，选择 SOF 文件 top. sof（图 6-25）。

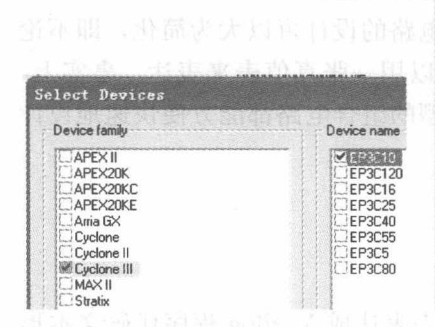


图 6-24 选择目标器件 EP3C10

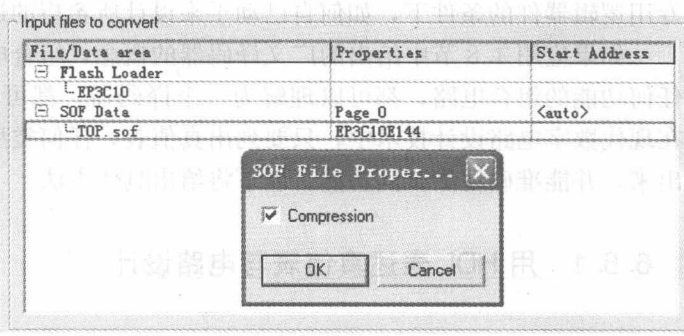


图 6-25 选定 SOF 文件后，选择文件压缩

为了使 EPCS4 能为今后可能的应用（如 SOPC 的 C 程序代码存放）腾出空间，需要压缩后进行转换。所以首先单击选中 Input files to convert 栏中的 top. sof 文件名，然后单击右下的 Properties 按钮，在弹出的对话框中选中 Compression 复选框（图 6-25）。最后单击 Generate 按钮，即生成所需要的间接编程配置文件。

(2) 下载 JTAG 间接配置文件。选择 Tool→Programmer 命令，选择 JTAG 模式，加入 JTAG 间接配置文件 EPCS4_file. jic，如图 6-26 所示作必要的选择，单击 Start 按钮后进行编程下载。

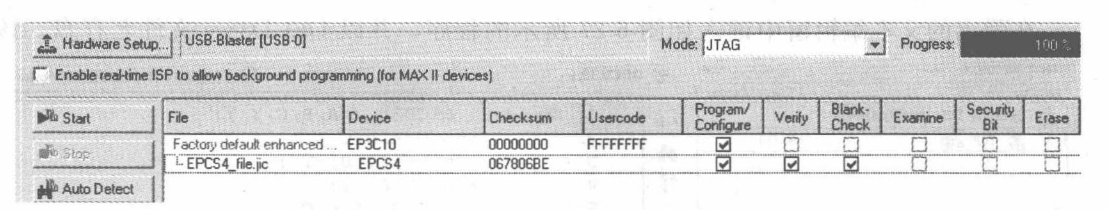


图 6-26 用 JTAG 模式经由 FPGA 对配置器件 EPCS4 进行间接编程

为了证实下载后系统是否能正常工作，在下载完成后，必须关闭系统电源，再打开电源，以便启动 EPCS 器件对 FPGA 的配置。然后观察 FPGA 内模块的工作情况。

另需注意，由于 FPGA 的输入输出端口 I/O 电平是 3.3V（包括下载于 FPGA 内部的 74 系列模块，如 74138 等），所以所有输入 FPGA 的信号的电平必须在这个范围内。如果信号来自传统 TTL 电平的 74LS 器件，必须在进入 FPGA 的信号通道上串接一个 200Ω 左右的电阻，且输入信号的频率越高，此电阻的阻值应该越小。若是输出，此类 3.3V I/O 电平的 FPGA 可以直接驱动 74LS、74HC、CD4000 等系列的逻辑器件。

6.5 用HDL来表述广义译码器

现代数字电路系统设计中，已不再使用诸如74系列等具有现成功能的标准逻辑器件来构建数字系统了（在端口驱动、三态总线控制等外围电路中仍有部分应用）。几乎所有逻辑功能模块，包括所有组合电路模块，如加法器、译码器、比较器等都必须由工程师自己来设计，并将设计好的所有功能模块集成于芯片中。本节将讨论，在不使用任何标准的专用逻辑器件的条件下，如何自己动手来设计所希望的逻辑功能模块。

如果应用4.8节中给出的广义译码器的概念，组合电路的设计可以大为简化，即不论任何功能的组合电路，都可以理解为一个译码器，都可以用一张真值表来表达。事实上，在现代数字电路设计技术中，只要列出真值表，任何类型的组合电路都能方便快速地设计出来，并能准确地测试其功能。以下将给出具体方法。

6.5.1 用HDL表述真值表与电路设计

利用Quartus II处理真值表的最好方法是先将真值表表达成Verilog程序代码文本形式。考虑到直接学习Verilog的语言规则和编程技术会花费太多的课时，目前也没有这个必要。最有效的方法是给出一个针对真值表的Verilog表达程序，仅将此程序的表述看成是一种特殊的表格，即特殊的真值表的表达形式，直接用来设计。即将学习的重点放在电路功能设计与实现方面，而暂时忽略对该表达格式或程序规则细节的理解。

1. HDL表述

打开Quartus II，打开以上6.3节的示例工程top。选择菜单File→New。在New窗口（图6-3）的Design Files页中选择Verilog HDL File的HDL文本编辑器。

在弹出的文本编辑窗中键入如图6-27所示的程序，并以DECD38.v文件名存盘。该

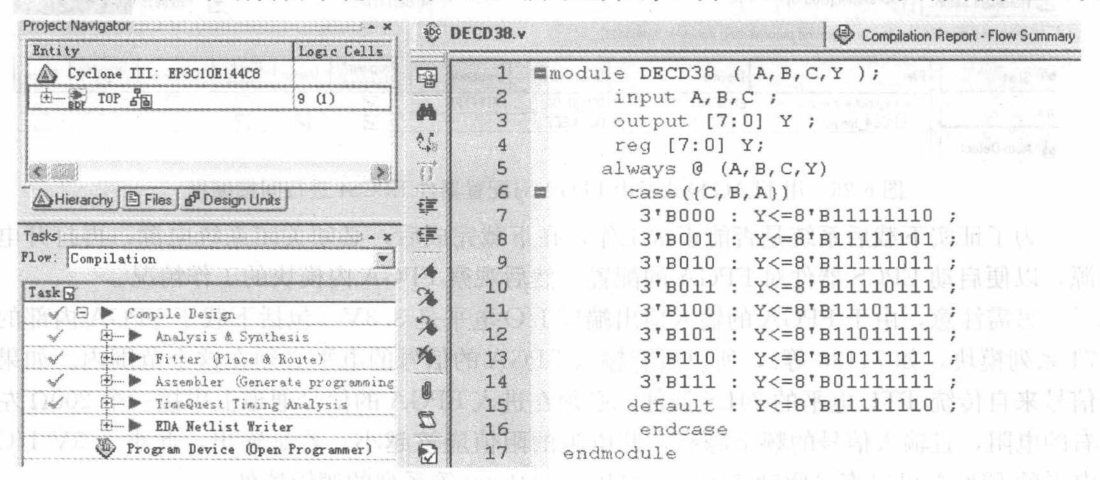


图6-27 3-8译码器真值表对应的HDL的case语句表述

程序实际上就是表4-9（输出都取反）的真值表的Verilog的一种表述。事实上，如果有了真值表，就可以模仿图6-27右栏的格式，将对应的数据填入到图6-27所示的“表”中。此“表”的其他表述可以基本保持不变。

对于图6-27，在编辑中需要注意如下几个问题：

（1）在Verilog HDL File文本编辑窗中编辑程序时，必须关闭中文输入工具，须在纯英文编辑环境中编辑。因为HDL综合器不认识中文标点符号，如逗号、分号等。

（2）第1行中的module是关键词，是电路模块的引导词，它的结尾词是第17行的关键词endmodule，凡是关键词必须小写，照抄。module和endmodule构成了一个“括号”，在它们之间的内容是对电路的描述。module右侧DECD38称为实体名，是设计者为当前设计取的名，分大小写，且用英语字母表示。这个名很重要，因为当前此文件存盘的文件名必须是DECD38.v，即存盘文件名必须与实体名一致，且后缀是.v。此名改动后，存盘文件名也必须随之改动。

第1行右侧括号中的文字（A,B,C,Y）是此电路模块所有的端口信号名。

（3）第2行用关键词input定义了此译码器模块的输入信号A、B、C。

（4）第3行用关键词output定义了此译码器模块的输出信号Y。请注意，在Y的左侧多了表述［7:0］，这表明Y是一个8位数据端口。这是一种矢量表达方式，即Y[7:0]。对于电路，也可称为总线表达方式；等效于定义了8个1位输出口，它们分别对应Y[7]、Y[6]、Y[5]、Y[4]、Y[3]、Y[2]、Y[1]、Y[0]。对于多个同类信号，矢量表达方式比较方便。当然也可以单独定义，如output Y7、Y6、Y5、Y4、Y3、Y2、Y1、Y0。

（5）第4行是定义输出信号的类型，reg是关键词，与output定义方式相同。一般而言，此类情况下，凡是输出信号都可按output表述照抄，即作对应的reg定义。

（6）第5行中，关键词always @可照抄，右侧括号中的内容称为敏感信号，可以将此模块的输入信号名都列于其中，列多了也不算错。每一信号名用逗号分开。

（7）第6行是表述真值表的case语句开始句。关键词case和第16行的endcase也构成了一个“括号”，在它们之间的内容是对“真值表”数据的表述。case右侧的语句是({C,B,A})，外边是小括号，里边是大括号。其中的{C,B,A}是输入信号数据的合并，即大括号有合并数据的功能。例如，若C=0、B=1、A=1，则{C,B,A}=3′B011。3′B011是Verilog的二进制数表示法，其中的3′表示此数有3位，B表示右侧的数据是二进制数。所以这时的{C,B,A}就等于二进制数011。

（8）第7～14行可以看成是真值表的表述句。以第8句为例，此句的含义是：当{C,B,A}=001时，即当C、B、A分别输入0、0、1时，Y[7:0]将输出11111101。即Y[7]、…、Y[1]、Y[0]分别输出1、…、0、1。在这里，符号“:”表示“于是”的意思；符号“<=”表示向Y输出数据的意思。

（9）第15行的default是关键词，它构成一个固定语句，必须放入。此句表示，当{C,B,A}不取以上任一数据时的默认输出操作。

2. 将Verilog文本表述转化为电路元件

在完成了图6-27的文件编辑和存盘后，就要将其变成一个电路元件以备调用。方法

如图 6-28 所示，选择菜单 File→Create/Update→Creat Symbol Files…。这时 Quartus II 首先对文件进行检查，如果没有错误，就会将其转化为一个元件，放在当前工程库中，即放在文件夹 d:\MY_PROJECT 中。

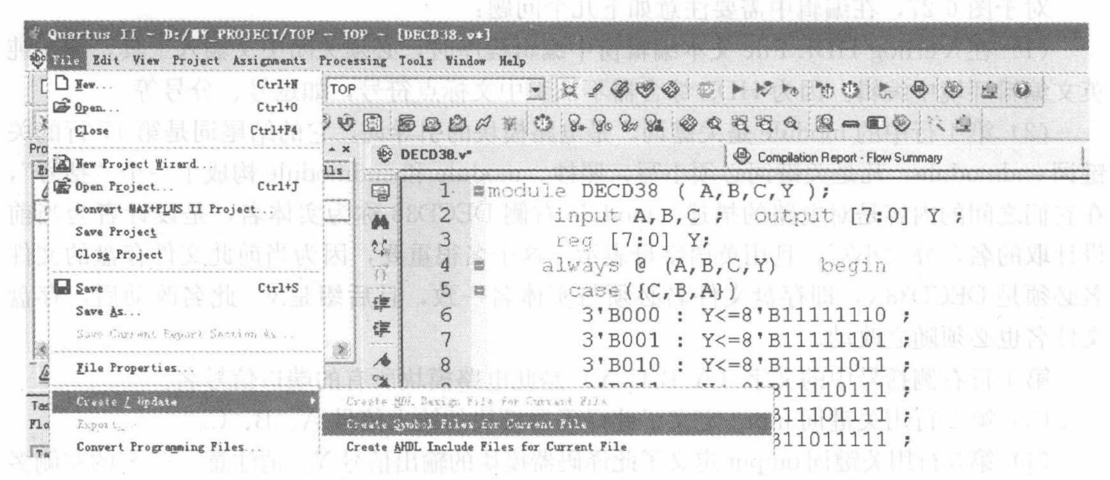


图 6-28　将 HDL 程序变成一个可以调用的原件模块

3. 完成电路设计

双击原理图编辑窗的任何空白位置，即能打开元件调用窗。单击左上 Libraries 栏的 Project。里面将出现已转换好的元件的元件名 DECD38（图 6-29）。按 OK 按钮，将此元件调入原理图（图 6-2）。然后删去图中原有的器件模块 74138，用 DECD38 取代之。连线时注意用信号标号来连接。如图 6-30 所示，在 DECD38 的输出端必须标上 Y[7..0]，其中是两个点，它表示输出有 8 根线。

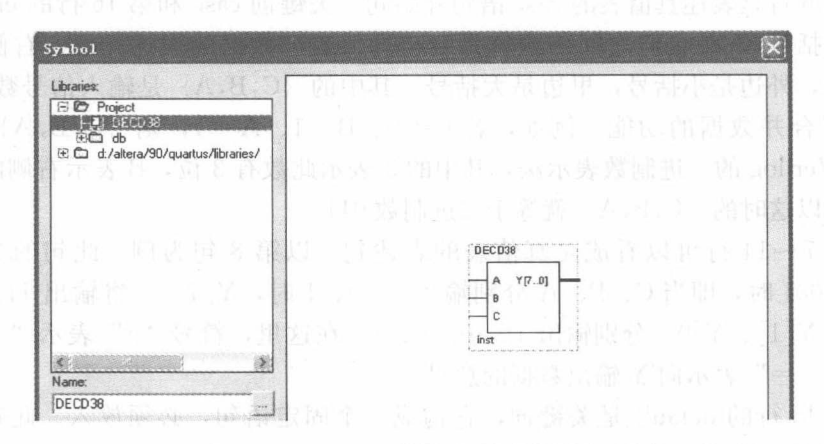


图 6-29　选择已生成好的元件 DECD38

注意：*在原理图表示中，矢量表述是 Y[7..0]，而在 Verilog 程序中的表述是Y[7:0]！*

现在，图 6-30 与图 6-2 的电路功能完全相同，只是图 6-2 中电路的译码器 74138 内部

是用逻辑门元件构建的，而图 6-30 的译码器 DECD38 是用 Verilog 文本表述的，原理图 6-30 是一种混合表述方式，即电路原理图和 Verilog 文本代码的混合表述形式。这种表述方式的优点是直观方便，且兼顾了两者的优势。

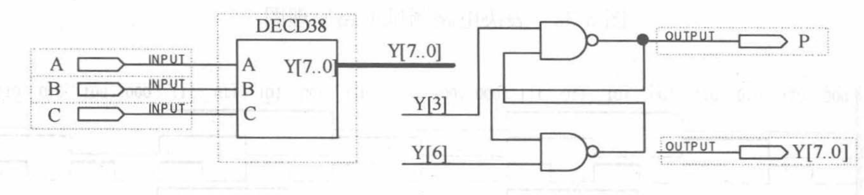


图 6-30 用 HDL 文本表述的 3-8 译码器 DECD38 连接好电路

4. **逻辑功能测试**

与对图 6-2 电路的处理相同，也要对图 6-30 的电路进行测试，测试方法也相同。首先是进行全程编译，然后建立仿真波形文件。由于输入输出端的名称并没有变化，因此可以用原来的 VWF 文件进行仿真。由于元件 DECD38 的功能与 74138 相同，所以仿真波形也应该一样，即电路图 6-30 的仿真波形图应该与图 6-15 相同。

6.5.2 3 人表决电路的 HDL 表述方式

再举一例，即利用 case 语句设计 3 人表决电路（参考第 4 章例 4-3 和表 4-3）。此项设计示例更具组合电路设计的一般性。

图 6-31 是表 4-3 的 case 语句表述，将其变成元件后构成的电路如图 6-32 所示。图 6-32 是在 Quartus II 的原理图编辑窗设计的表决电路。建立对应的工程，选择目标器件后，即可对此电路进行全程编译和仿真测试。仿真波形如图 6-33 所示，对照表 4-3，可以判定，此项设计是正确的。

```
1  module  JG3 (A,B,C,X,Y);
2     input A,B,C; output X,Y ;
3        reg X,Y ;
4    always @(A,B,C,X,Y)
5     case ( {A,B,C} )
6       3'B000 : begin X<=0; Y<=1; end
7       3'B001 : begin X<=0; Y<=0; end
8       3'B010 : begin X<=0; Y<=0; end
9       3'B011 : begin X<=0; Y<=0; end
10      3'B100 : begin X<=0; Y<=0; end
11      3'B101 : begin X<=1; Y<=0; end
12      3'B110 : begin X<=1; Y<=0; end
13      3'B111 : begin X<=1; Y<=0; end
14      default : begin X<=1; Y<=0; end
15    endcase
16 endmodule
```

(a)

```
1  module  JG3 (A,B,C,X,Y);
2     input A,B,C; output X,Y ;
3        reg X,Y ;
4    always @(A,B,C,X,Y)
5     case ( {A,B,C} )
6       3'B000 : {X,Y} <=2'B01;
7       3'B001 : {X,Y} <=2'B00;
8       3'B010 : {X,Y} <=2'B00;
9       3'B011 : {X,Y} <=2'B00;
10      3'B100 : {X,Y} <=2'B00;
11      3'B101 : {X,Y} <=2'B10;
12      3'B110 : {X,Y} <=2'B10;
13      3'B111 : {X,Y} <=2'B10;
14      default : {X,Y} <=2'B10;
15    endcase
16 endmodule
```

(b)

图 6-31 表 4-3 对应的两种 case 语句的表述方式

仿真波形图（图 6-33）中对 3 个输入信号的编辑使用了 group 合并功能。这样一来，在设置输入数据时就可以使用曾经用过的“C”功能来连续设置了。

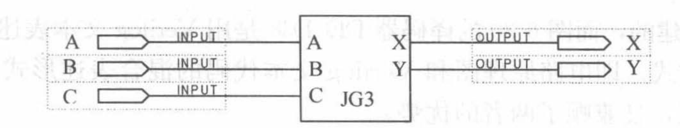


图 6-32 表决电路测试电路原理图

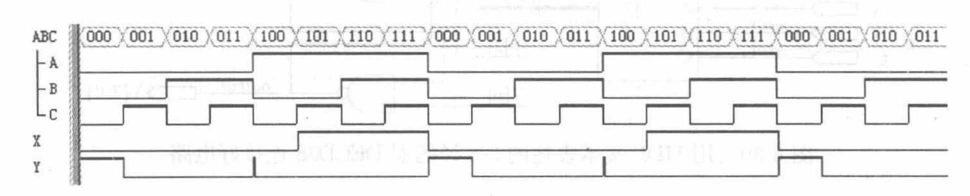


图 6-33 3 人表决电路的仿真波形

可以注意到波形图（图 6-33）中的输出有毛刺，这是输入数据在硬件电路中的延时造成的，原因是 011 至 100 的逻辑变化最大，导致的电路延时时差也最大。由于此毛刺脉冲宽度很小，对实际结果没有影响。当然，这种毛刺并非总会出现，如果选择更高速度级别的 FPGA，这些毛刺就可能在仿真波形中消失。

另一方面，这张仿真波形图也说明，通过 Quartus II 的时序仿真，可以获得反应逻辑设计硬件电路真实性能的波形图。显然，类似图 6-33 的毛刺在传统的逻辑分析方法下是不可能获得的。

比较程序图 6-27 和图 6-31（a），会发现图 6-31 的程序多了一组 begin_end 表述。表述 begin_end 也是关键词，称为块语句。块语句本身没有什么功能，此语句只相当于一个“括号”，在此括号中的语句都被认定归属于同一操作模块，统一操作。例如，对于第二条赋值语句：3′B001：begin X<=0；Y<=0；end，其含义是当 ABC=001 时，X 输出 0，Y 输出 0（X 和 Y 同时输出 0，没有先后）。

Verilog 要求，凡是在 case 语句中，输出信号多于一组的情况，必须加上 begin_end 表述将其“括”起来。图 6-27 的程序由于只有一组输出，故可省去 begin_end 表述。

图 6-31（a）和图 6-31（b）所描述的功能完全相同，只是图 6-31（b）中用了表述简洁的具有并位功能的大括号 {}。这在图 6-27 的程序中已经用过了。

通过对以上二示例的学习和实践，读者应该初步学会了如何将一张表征特定逻辑功能的真值表表述成 Verilog 代码，以及用 Quartus II 验证其正确性的方法。

6.5.3 用 HDL 对真值表的其他表述方式

以上的示例表明，用 case 语句来表示描述组合电路的真值表或者说广义译码器十分方便，而且还很灵活，这使得组合电路的设计变得非常容易。以下再给出两则描述组合电路的不同形式的 case 语句表述形式，读者可以根据自己的设计需要，灵活套用。

1. 文字表达方式的多路选择器设计

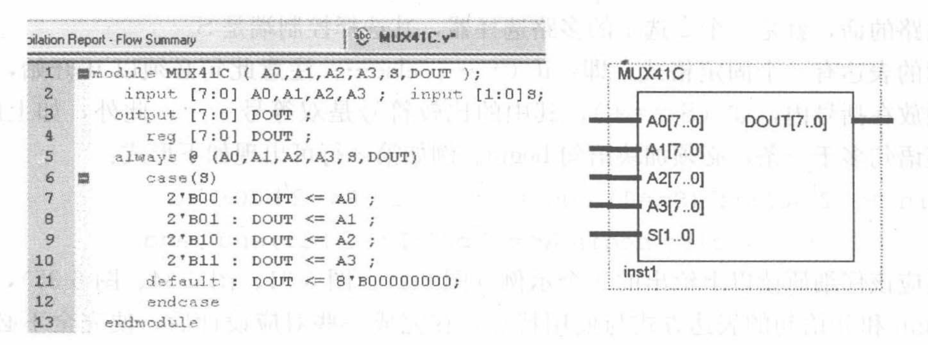
上节的示例表明，需要将输入输出信号中所有可能出现的数据都列于真值表中，然后

用case语句表述出来。这种方式的好处是直观，但缺点是所能表达的组合电路的规模太小，实用意义不大。例如以上的表决电路只涉及3个人，没有实用意义。如果表决的人多达30人、300人，其真值表就将大到无法用简单的真值表来表达。因此在实际设计中使用文字表达将要高效得多。

例如要设计一个8位4选1型多路选择器（每一通道是8位数据宽度），如果用以上的方法，首先列出此电路输入输出所有可能的二进制数据的真值表，然后用case语句描述此真值表，理论上也可以完成设计，但实际实行起来就很不容易。

如果按图6-34（a）给出的case语句表述方法就十分简洁。程序中，当通道控制信号S分别被输入00、01、10、11时，8位输出端口BOUT将分别输出来自A0、A1、A2、A3端口的8位数据。当然用此法设计多路多位数据分配器也是相同形式。

用图6-28的选项可以将图6-34（a）的代码表述方式转换为图6-34（b）的原理图符号。




```
1  module MUX41C ( A0,A1,A2,A3,S,DOUT );
2    input [7:0] A0,A1,A2,A3 ;  input [1:0]S;
3    output [7:0] DOUT ;
4      reg [7:0] DOUT ;
5    always @ (A0,A1,A2,A3,S,DOUT)
6      case(S)
7        2'B00 : DOUT <= A0 ;
8        2'B01 : DOUT <= A1 ;
9        2'B10 : DOUT <= A2 ;
10       2'B11 : DOUT <= A3 ;
11     default : DOUT <= 8'B00000000;
12       endcase
13 endmodule
```

(a)8位4选1型多路选择器的case语句描述　　(b)由图(a)代码生成的原理图元件符号

图6-34　8位4选1型多路选择器的“真值表”描述及转换后的原理图元件模块

2. 含有条件判定情况的真值表的case语句表述

一般的真值表的输入输出数据一开始就是确定的，且直接写在表中。但有的组合电路需要根据外部的控制信号，在相同的输入数据情况下，作不同的输出。图6-35就是一个这样的程序（对应的电路模块如图6-36所示），它有很多实际应用。

图6-35程序的仿真波形如图6-37所示。图6-35第8行的含义如下：

当输入端DIN[1:0]=01时，DOUT输出010，且若选择控制输入端S=1（高电平），则DATA口输出110，否则（即S=0），DATA输出011。显然，把这里的if语句

```
1  module SM ( S,DIN,DATA,DOUT );
2      input S ; input [1:0] DIN ;
3      output [2:0] DATA,DOUT ;
4      reg [2:0] DATA,DOUT;
5    always @ (S,DIN,DATA)
6      case( DIN )
7        2'B00 : begin DOUT<=3'B111; DATA<=3'B001 ; end
8        2'B01 : begin DOUT<=3'B010; if (S==1'B1) DATA<=3'B110; else DATA<=3'B011 ; end
9        2'B10 : begin DOUT<=3'B111; DATA<=3'B000 ; end
10       2'B11 : begin DOUT<=3'B111; DATA<=3'B111 ; end
11     default :       DOUT<=3'B111;
12       endcase
13   endmodule
```

图6-35　含条件判断情况的“真值表”表达样本

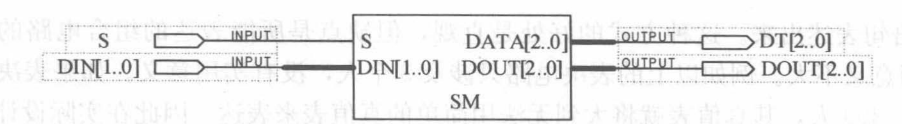


图 6-36　对应图 6-35 程序的测试电路原理图

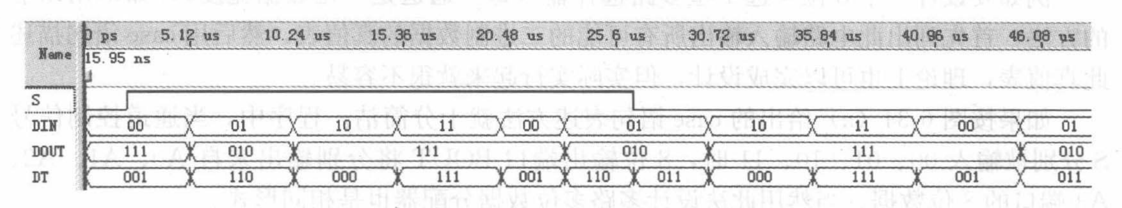


图 6-37　图 6-36 电路的仿真波形

翻译成电路的话，就是一个 2 选 1 的多路选择器，其选择控制端是 S。

此句的表达有一个固定格式，即：if（）…；else…；注意此句必须以 if 开始，条件表述必须放在括号内，如（S==0），其中的比较符号是双等号==；此外，如上所述，如果赋值语句多于一条，必须加块语句 begin。例如第 8 行可出现如下形式：

```
begin D<=2'b01;if (S==1) begin A<=2'b10;B<=2'b00;end
                        else begin A<= 2'b01;B<=2'b11;end ;end
```

读者应该仔细研读以上给出的 4 个示例（图 6-27、图 6-31、图 6-34、图 6-35），熟悉 case、begin 和 if 语句的表达方式与使用特点，在完成一些对应设计后，便完全不必再去专门学习 Verilog 语法，即可迅速掌握用 Verilog 程序代码描述和设计任何形式的组合电路的方法。需要牢记的是，无论用 case 语句如何表达，也无论用电路原理图如何描述，对其实际逻辑功能的判断绝不能仅停留在语句描述的判断或电路结构的分析上，因为这只能作为一种极初步的定性判断，而真正重要的是必须通过时序仿真来确定设计模块的逻辑功能和时序特性！这是现代逻辑设计有别于传统手工设计的一个十分重要的方面。

以后，凡是遇到组合电路设计，最快捷的方法就是根据以上提供的方法来完成。再次提醒，根本捷径就是机械地套用！即只关心使用这种表述格式去设计电路，而不要去深究为什么如此表述（表述得正确与否仿真一次即明白）。这可以为你节约很多时间，因为以后会有机会去理解和学习的。

以上的一些格式已包括了几乎所有组合电路的设计。现代数字系统的设计其实更注重结果，即高效高性能的电路产品，而不必计较用什么方法、什么工具、什么流程得到的，自主创新更没有常规的流程可循，此所谓“道可道，非常道”。

在后面介绍的时序电路自动设计中，将更多地利用本节介绍的方法。

6.6　数字方法去抖动和延时电路设计

本节将结合以上介绍的 Quartus II 原理图输入设计方法和第 5 章有关触发器的知识，介绍简单但实用的数字去抖动电路和数字延时电路的设计和分析方法。

6.6.1 数字去抖动电路设计

曾经在 5.2.4 节介绍了一个利用 RS 触发器去除机械电子抖动的电路。这个电路(图 5-6)的缺点是：

(1) 键信号必须在 R、S 两个点输入，而无法只从一个输入口进入，所以有了许多应用限制。

(2) 键脉冲宽度只能靠手动按键决定，无法由电路控制决定，所以只能适用于慢速的键抖动去除，而无法对任意宽度含随机抖动信号进行处理。

作为触发器的实用示例，这里将介绍一种使用 D 触发器构成的电路，能去除含电子抖动的任意形式及几乎任意频率的信号，且能从电路上控制输出信号的脉宽。这是一种更实用、功能更完善的电路。这种电路本质上就是滤波器，它可以将信号中的毛刺、随机噪声信号或电子抖动信号都“滤除”，只让真正的数据信号通过此电路。

在介绍此电路之前，首先介绍有关脉冲波形参数、信号频率和周期的概念。

1. 脉冲参数和信号频率概念

在数字系统中，除了固定的高低电平信号，就是被抽象成方波的时钟脉冲。这些在 0、1 两种状态间变化的脉冲波形，常被抽象为严格的矩形波。但在现实的情况下，并非如此。因为当方波信号在传输了一段距离后，由于信号通道上传输介质介电常数的不一致性(导致对不同频率的散射效应)、介质损耗、通道上分布电容和分布电感的影响等，使得实际数字系统中的脉冲波形发生了变形，大致情况如图 6-38 所示。脉冲的上升沿和下降沿呈一定的“坡度”，且这些外部因素越明显，则脉冲的“坡度”越长。如果用示波器对这些脉冲信号进行观察，可以清晰地发现，这些所谓的矩形波脉冲的边沿并非是垂直的，即并非单纯在 0、1 两种状态之间变化，它们在 0、1 状态之间有一定的过渡状态。当然，若信号频率不高，传输距离不长，且通过的介质简单等情况下，为了简化讨论，仅将注意力集中于信号的处理和数据传输上，则将脉冲当成严格的方波来对待。

为了便于定量表述和分析，可以根据图 6-38，用下列几种波形参数，对不同特性的脉冲进行区分和表述。

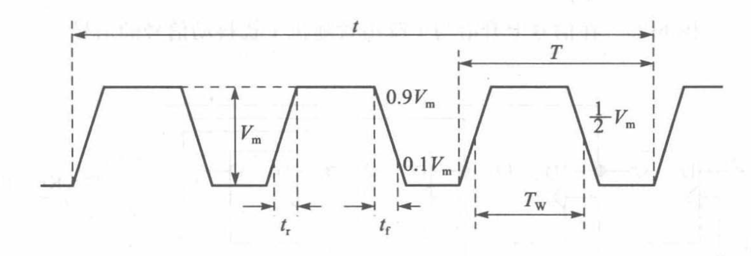


图 6-38 脉冲波形概念和参数说明图

脉冲周期 T：在周期性重复的脉冲序列中，两个相邻脉冲波形之间的时间间隔，称之为脉冲周期。即若单个方波信号所需的时间是 T 秒，则称此信号的周期是 T。

脉冲频率 f：单位时间内的脉冲个数，称之为脉冲频率，可以由脉冲周期直接获得频率值，可用 $f=\frac{1}{T}$ 这个公式进行计算。可以这样来考虑，如果有一个周期信号波形发生器，其输出的周期信号波形在每秒钟内的个数有 N 个，则称此信号的频率是 NHz。图 6-38 显示了含有 3 个完整的周期性方波信号的波形。如果如图 6-38 所示的 t 时刻中包含有 $N=3$ 个完整脉冲周期，则此信号的频率 $f=N/t$。如果 $t=1$s，则称此信号的频率 f 是 3Hz。显然，频率与周期的关系是倒数关系：$f=N/t=3/3T=1/T$。

脉冲幅度 V_m：脉冲波形的峰峰值，即变化的最大幅度，称之为脉冲幅度。

上升时间 t_r：脉冲电压从 $0.1V_m$ 到 $0.9V_m$ 的变化时间，称之为上升时间。

下降时间 t_f：脉冲电压从 $0.9V_m$ 到 $0.1V_m$ 的变化时间，称之为下降时间。

脉冲的上升沿时刻定义为脉冲电压从低电平到高电平变化，在 $\frac{1}{2}V_m$ 这个时刻为上升沿时刻；同样，脉冲的下降沿时刻定义为脉冲电压从高电平到低电平变化，在 $\frac{1}{2}V_m$ 这个时刻为下降沿时刻。

脉冲宽度 T_w：从脉冲的上升沿时刻到脉冲的下降沿时刻间的时间间隔，称之为脉冲宽度，一般简称为脉宽。

占空比 q：脉冲为高电平的时间与为低电平的时间比值，称之为占空比。在实际应用中，占空比有两种表示方式。以方波为例，方波的占空比可以写成 1∶1，也可以写成 50%，两种写法都可以。即 $q=T_w/(T-T_w)$ 或者 $q=\frac{T_w}{T}100\%$。

2. 数字去抖动电路设计

如图 6-39 所示的模拟信号波形，在正常信号的上升沿和下降沿处有一些随机干扰信号，类似于一些毛刺脉冲群，或随机抖动脉冲。为了去除这些抖动干扰脉冲，可以使用如图 6-40 所示的电路来实现。

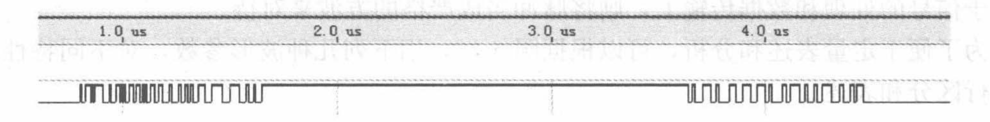


图 6-39 在信号上升沿与下降沿含随机干扰抖动信号的信号

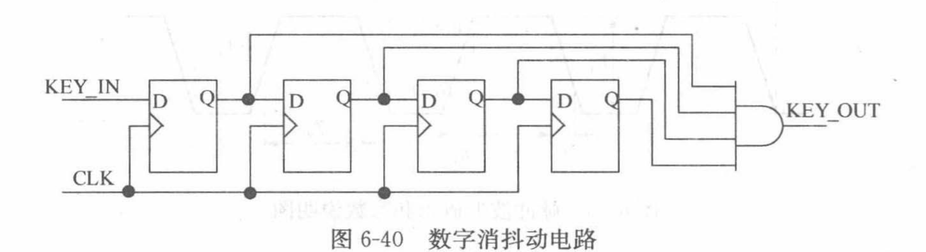


图 6-40 数字消抖动电路

图 6-40 所示的电路由 4 个 D 触发器和 1 个 4 输入与门构成。电路有一个工作时钟

CLK。4 个 D 触发器连接成同步时序方式，即将它们的时钟输入端都连在一起。工作时与时钟同步工作，输入信号以移位串行方式向前传递。其信号输入口是 KEY_IN，输出口是 KEY_OUT。

分析此电路可以发现，其“滤波”功能的关键是这样的，当信号被串入电路后，能在 KEY_OUT 输出脉冲信号的条件是，必须在 4 个 D 触发器的输出端 Q 都同时为 1，此与门才输出高电平。由于干扰抖动信号是一群宽度狭窄的随机信号，在串入时，很难十分整齐地同时使与门输出为 1，而只有正常信号才有足够的宽度通过此电路，从而起到了“滤波”的功能。

3. 时序仿真

图 6-41 是对图 6-40 电路的时序仿真波形。

在编辑输入的激励信号时要作一些设置：首先建立如图 6-41 所示的 vwf 仿真波形图。为了设置工作时钟 CLK 的输入信号，先单击图 6-41 中 CLK 的左侧，使 CLK 的整个仿真时间区域变成蓝色，然后选择单击波形编辑窗左侧的时钟设置符号，一个小“钟”。这时弹出的对话框如图 6-42 所示。首先选择 CLK 的周期：在 Period 栏设置 CLK 的周期为 600ns。注意此前已经设置整个仿真时间区域是 55us（微秒）。时钟波形偏移量 Offset 选择默认。最下方是占空比设置，已默认为 50%，即 Duty cycle（%）：50。

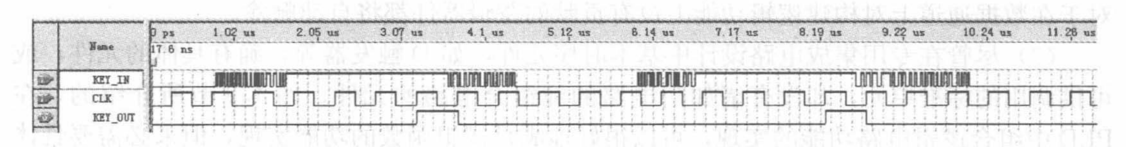


图 6-41 消抖动电路仿真波形

第二个需要编辑设置的是输入信号 KEY_IN。为了得到比较符合实际情况的输入波形，必须用鼠标仔细编辑绘制。为此，首先选择工程管理窗的 View 菜单项，在出现的下拉菜单中（图 6-43）单击 Snap to Grid。此项选择的目的是，当鼠标在输入信号 KEY_IN

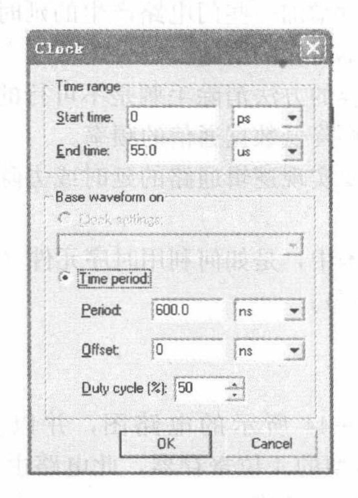


图 6-42 设置时钟周期



图 6-43 关闭分格限制

时间域上时，可以随心所欲地绘制随机分布的任意宽度的脉冲。

编辑好输入的激励信号后就可以启动仿真器了。仿真波形即如图 6-41 所示，波形显示，输出信号十分干净，已滤除了所有干扰脉冲信号。但也不难发现存在如下情况：

(1) 输出的脉宽变小了，它只等于 CLK 的一个周期宽度，且输出信号比原来的信号有一个明显的延迟。显然 CLK 的频率可以决定输出信号的脉宽和延迟量。

(2) CLK 的频率要足够高，即至少应该有 4 个上升沿被包含在正常信号脉宽中，否则所有信号都无法通过此电路。

(3) CLK 的频率不能太高，即其周期不能太小于干扰信号的脉宽。

(4) 如果增加 D 触发器的数量，可以在一定程度上提高滤波性能。

6.6.2 数字延时电路的设计与测试

为了消除数字电路中的冒险竞争现象，或使波形有微量的延时，传统数字电路设计技术中较常用的方法是在通道上增加门电路，或利用冗余技术来解决，甚至使用并接滤波电容的手段。但是这些方法在基于计算机自动设计的现代数字系统设计技术中都是行不通的，这是因为：

(1) 在自动设计过程中，EDA 软件只负责按照设定的约束条件进行综合与优化，它对于在数据通道上对构建逻辑功能上没有贡献的逻辑器件都将自动删除。

(2) 尽管在专用集成电路设计中基本时序元件，如 D 触发器等，确有具体的元件，或可供调用的标准单元，却没有诸如门电路的纯组合电路的元件。如 4.9 节所介绍的，在 PLD 中组合逻辑电路功能的实现，可以很好地满足逻辑函数的功能实现，但未必需要具体的门电路来实现；门电路逻辑功能的实现常常可能只是在可编程门阵列中多一个或少一个熔丝点而已。因此，在逻辑电路图中逻辑门的存在不一定会影响延时。而且有时会起到相反的作用，即在逻辑电路图中增加逻辑门有时甚至会减少延时！

(3) 由于每一种目标器件的基本延时特性都是不同的，且延时特性会随着外部因素，如温度，气压的变化而变化，因此如果只是希望通过增加一些门电路产生的延时来克服冒险竞争，其延时量极难控制，显然这本身就是一个不可靠的措施。

(4) 在高频率信号传输的条件下，通过外接电容的方法消除毛刺是不可行的，因为这些电容会将毛刺连同正常信号一同旁路掉，从而大幅降低数据通信的频率。

在现代数字系统设计技术中，为了某种目的需要实现逻辑通路的延时或去除毛刺信号时，是决不会考虑使用组合电路来实现的。

本例希望通过一个简单示例说明在现代数字技术中，是如何利用时序元件（如 D 触发器）来延时的（在一定调节下可以利用延时去除毛刺）。

1. 设计一个库元件

首先建立一个原理图工程，然后设计如图 6-44 所示的电路图，并以文件名为 DFF4. bdf（文件名可以随意取）存盘。这是一个典型的 4 位寄存器，此电路由 4 个 D 触发器组成。D 触发器的元件名是 DFF，键入名字后即能调出。

为了将此电路变成一个功能元件，以备在其他电路中调用，可以利用元件符号生成功能选项。方法可参考图 6-28，即在 File 选项中的下拉菜单中选择 Create/Update 项，然后选择对应的下拉菜单，单击 Create Symbol Files for Current File 项。之后，图 6-44 的电路就变成了一个以此 DFF4 为元件名的元件符号，并存在当前的文件夹中。

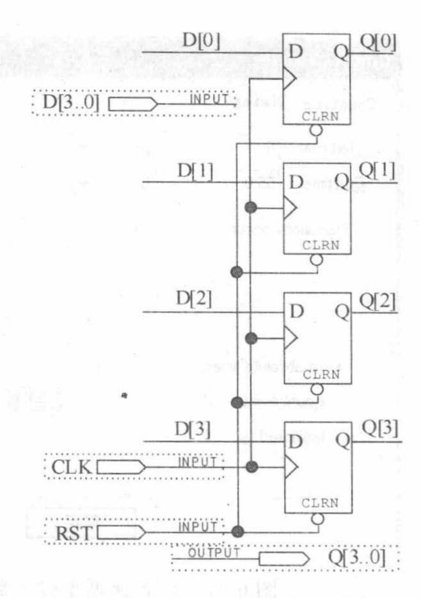


图 6-44　DFF4 4 位寄存器电路

2. 设计顶层电路

创建一个顶层原理图工程，然后在弹出的类似图 6-7 所示的对话框中键入刚才设计好的元件名 DFF4，即能将此元件调入原理图编辑窗中。然后按照图 6-45 所示的电路完成电路系统的设计。这个电路由 3 个 4 位寄存器 DFF4 组成，如果双击元件 DFF4，将弹出此元件内部电路结构。在仿真前，对此工程进行全程编译。

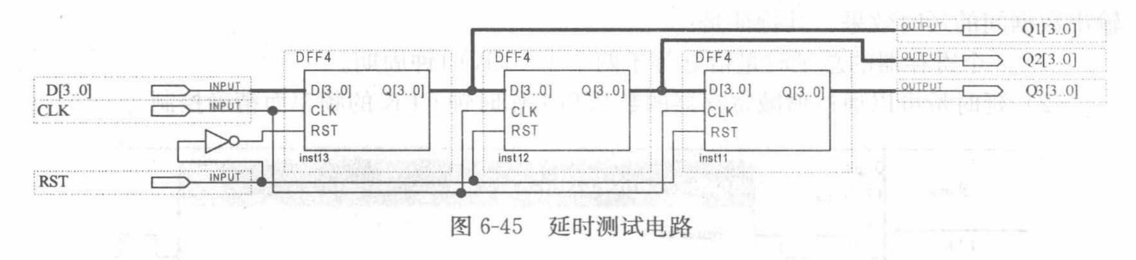


图 6-45　延时测试电路

3. 时序仿真

首先建立 vwf 仿真文件和拖入仿真信号。为了设置输入信号的激励波形，在如图 6-46 所示的 vwf 编辑窗中首先单击输入数据 D，使之变成蓝色；再选择左侧工具栏的“C”按

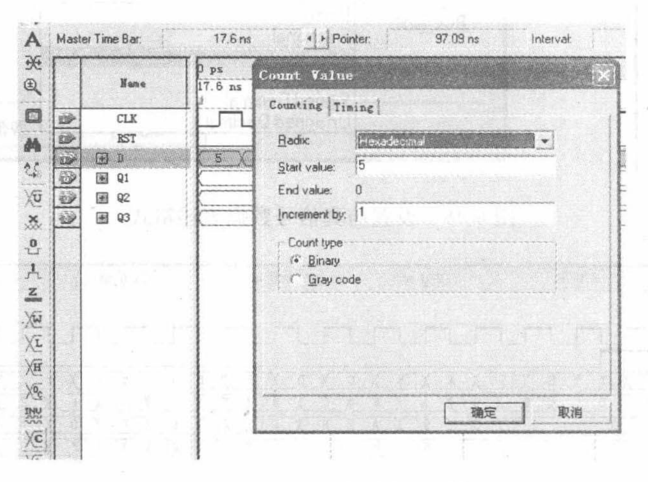


图 6-46　设置仿真用输入数据

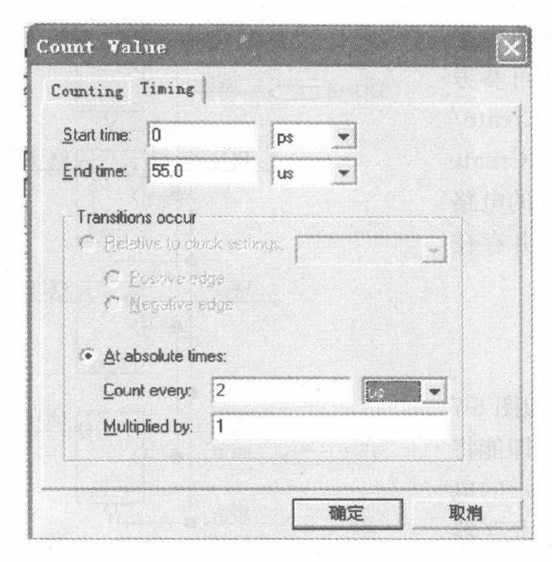


图 6-47 设置递增型输入数据时间间隔

钮，在随后弹出的对话框（图 6-46）中选择数据类型为十六进制格式 Hexadecimal；起始值为 5（Start value）；递增为 1（Increment by）；选择计数模式为二进制 Binary。

然后单击图 6-46 的 Count Value 窗的 Timing 页。在此页中（图 6-47）的 Count every 栏选择 2us（微秒），表示每 2us 递增一个数值。

为了方便数据显示，最后要对输出数据 Q1、Q2、Q3 的格式作变换。先双击图 6-48 波形编辑窗的 Q1 的名字。在弹出的对话框（图 6-48）的 Radix 栏，选择十六进制数据格式 Hexadecimal；此后 Q2、Q3 也作同样选择。启动仿真后可以得到如图 6-49 所示的时序仿真波形。从波形中可以看到输入与输出数据间的延时效果。其特征是：

（1）3 个寄存器的总延时量恰好等于两个半 CLK 时钟周期。

（2）延时量可以通过增减寄存器的数量和工作时钟 CLK 的频率而精确控制。

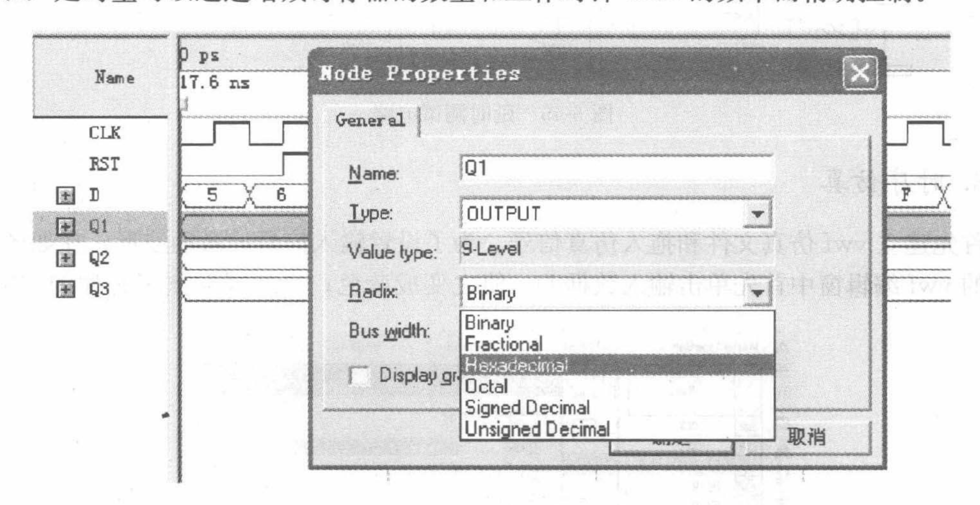


图 6-48 设置仿真信号数据表述格式

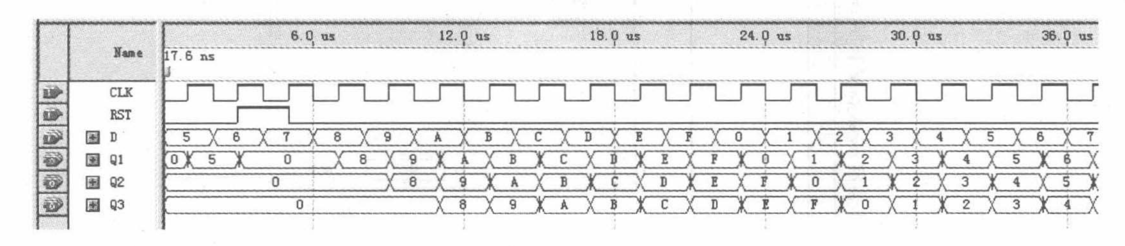


图 6-49 图 6-45 电路仿真波形

实 验

6-1 用 Quartus II 库中的宏功能模块 74138 和与非门实现指定逻辑函数

按照 6.3 节和 6.4 节的流程，使用 Quartus II 完成图 6-2 电路的设计，包括：创建工程，在原理图编辑窗中绘制此电路，全程编译，对设计进行时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译，编程下载于 FPGA 中进行硬件测试。最后完成实验报告。

6-2 用两片 7485 设计一个 8 位比较器

用两片 4 位二进制数值比较器 7485（Quartus II 库中的宏功能模块）串联扩展为 8 位比较器，使用 Quartus II 完成全部设计和测试，包括创建工程、编辑电路图、全程编译、时序仿真及说明此电路的功能、引脚锁定、编程下载，进行硬件测试。最后完成实验报告。

6-3 设计 8 位串行进位加法器

首先根据图 4-33，用半加器设计一个全加器元件，然后根据图 4-34，在顶层设计中用 8 个 1 位全加器构成 8 位串行进位加法器。给出时序仿真波形并说明之，引脚锁定编译，编程下载于 FPGA 中进行硬件测试。最后完成实验报告，讨论这个加法器的工作速度。

6-4 设计 8 位十进制数动态扫描显示控制电路

（1）根据电路图（图 4-22）利用 Quartus II，用 7448 和 74138 宏功能元件设计实现 8 位十进制数动态扫描显示控制电路，并在实验系统上控制 7 段数码管。位选信号 S2、S1、S0 可以用 3 个键控信号手动控制。给出时序仿真波形并说明之，引脚锁定，编程下载于 FPGA 中进行硬件测试。最后完成实验报告。

由于是在 FPGA 中设计该系统，可以直接输出驱动数码管。

（2）给出真值表，以上所有控制电路用同一 case 语句表达出来，然后硬件实现。

6-5 设计一个十六进制 7 段显示译码器

用 Verilog 的 case 语句设计一个可以控制显示共阴 7 段数码管的十六进制码 7 段显示译码器。首先给出此译码器的真值表，此译码器有 4 个输入端：D、C、B、A。D 是最高位，A 是最低位；输出有 8 位：p、g、f、e、d、c、b、a，其中 p 和 a 分别是最高和最低位，p 控制小数点。对于共阴控制，如果要显示“A”，输入 DCBA＝1010；若小数点不亮，则输出 pgfedcba＝01110111＝77H。给出时序仿真波形并说明之，引脚锁定，下载于 FPGA 中对共阴数码管进行硬件测试。

提示：参考程序如例 6-1 所示。用输入总线的方式给出输入信号仿真数据，仿真波形示例图如图 6-50 所示。

【**例 6-1**】十六进制 7 段显示译码器参考程序。

```
module LED7SEG (A, LED7S);
 input [3: 0] A;
 output [6: 0] LED7S;
 reg [6: 0] LED7S;
 always @  (A)
   case (A)
   4'B0000: LED7S< = 7'B0111111;
   4'B0001: LED7S< = 7'B0000110;
   4'B0010: LED7S< = 7'B1011011;
   4'B0011: LED7S< = 7'B1001111;
   4'B0100: LED7S< = 7'B1100110;
   4'B0101: LED7S< = 7'B1101101;
   4'B0110: LED7S< = 7'B1111101;
   4'B0111: LED7S< = 7'B0000111;
   4'B1000: LED7S< = 7'B1111111;
   4'B1001: LED7S< = 7'B1101111;
   4'B1010: LED7S< = 7'B1110111;
   4'B1011: LED7S< = 7'B1111100;
   4'B1100: LED7S< = 7'B0111001;
   4'B1101: LED7S< = 7'B1011110;
   4'B1110: LED7S< = 7'B1111001;
   4'B1111: LED7S< = 7'B1110001;
   default: LED7S< = 7'B0000000;
   endcase
 endmodule
```

Name:	Value:	5.0us	10.0us	15.0us	20.0us	25.0us	30.0us	35.0us	40.0us	45.0us	50.0us
A	B 0001	0000 0001	0010 0011	0100 0101	0110 0111	1000 1001	1010 1011	1100 1101	1110 1111	0000 0001	
LED7S	H 06	3F 06	5B 4F	66 6D	7D 07	7F 6F	77 7C	39 5E	79 71	3F 06	

图 6-50 仿真波形示例图

6-6 设计一个 5 人表决电路

用 case 语句设计一个 5 人表决电路，参加表决者 5 人，同意为 1，不同意为 0，同意者过半则表决通过，绿指示灯亮；表决不通过则红指示灯亮。给出时序仿真波形并说明之，引脚锁定，编程下载，硬件测试。最后完成实验报告。

第7章 时序逻辑电路的分析与设计

本章首先介绍传统数字技术中有关时序电路的分析与设计方法及逻辑器件的使用方法，使读者对时序逻辑电路设计技术的历史沿革有一个感性的认识。尽管这些分析和设计方法在现代数字产品设计中已不再直接使用，但掌握其中的基本概念、基本方法和基本原理，对于下一章中将要提出的更一般形式的数字电路模型和相关的分析与处理技术是十分有利的，而且在后续课程中也有助于深入理解基于现代电子设计技术的数字系统设计方法和优化技术。

本章基于传统的手工数字技术，将详细介绍时序逻辑电路的分析与设计方法，介绍寄存器、各种类型的计数器及其他实用时序电路的分析与设计方法。

7.1 时序逻辑电路的特点与功能

根据数字逻辑电路的特点，可以将其划分为组合逻辑电路和时序逻辑电路两大类。组合逻辑电路的特点是任意时刻的输出仅仅取决于当时的输入，而与电路过去的状态无关，即组合电路没有“记忆”功能。而时序逻辑电路在任意时刻的输出不仅与当时的输入有关，还与过去的状态有关，即时序电路具有“记忆”功能。

7.1.1 时序电路的结构

时序电路的一般结构如图 7-1 所示，其中 $x(x_1, x_2, \cdots, x_n)$ 是时序电路的输入信号，$y(y_1, y_2, \cdots, y_j)$ 是组合电路的输出信号，也是时序电路的激励信号；$z(z_1, z_2, \cdots, z_k)$ 是时序电路的输出信号，$w(w_1, w_2, \cdots, w_l)$ 是时序电路中的反馈信号，反馈到组合电路的输入端，与输入信号共同决定时序电路的输出状态。

这些信号之间的逻辑关系可以表述为

$$y_i = f_i(x_1, \cdots, x_n, w_1, \cdots, w_l) \quad (i = 1, 2, \cdots, j) \tag{7-1}$$

$$z_i = g_i(y_1, \cdots, y_j, w_1, \cdots, w_l) \quad (i = 1, 2, \cdots, k) \tag{7-2}$$

$$w_i = h_i(y_1, \cdots, y_j, w_1, \cdots, w_l) \quad (i = 1, 2, \cdots, l) \tag{7-3}$$

其中式（7-1）是存储电路的驱动方程或激励方程；式（7-2）是输出方程；式（7-3）是状态方程。在时序电路中，存储电路的状态通常是用触发器的状态表示的。在式（7-2）中用 z_1，…，z_k 表示每个触发器的现态，用 z_1^{n+1}，…，z_k^{n+1} 表示每个触发器的次态。

从时序逻辑电路的一般结构可以看出，时序逻辑电路有以下特点：

（1）时序电路由组合电路和存储电路共同组成，具有记忆过去状态的功能。

（2）时序电路中存在反馈回路。

（3）时序电路输出由电路当时的输入和电路原来的状态共同决定。

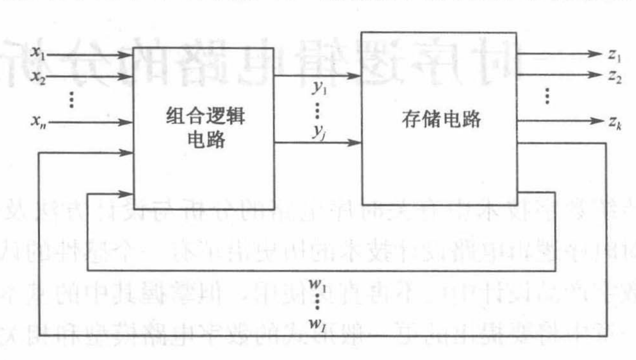


图7-1 时序电路的一般结构

7.1.2 时序电路的分类

通常可按电路的工作方式和电路输出对输入的依赖关系来对时序电路进行分类。如果按电路的工作方式分类，可将时序逻辑电路分为同步和异步时序电路。在同步时序逻辑电路中，所有的触发器的时钟连在一起，即在同一个时钟脉冲作用下，状态转换同步发生；在异步时序电路中，各触发器的时钟脉冲不相同，时钟输入相对独立，所以各触发器的状态转换不是同时发生的。如果按电路对输入的依赖关系分类，则又可分为Mealy型时序逻辑电路和Moore型时序逻辑电路，前者的输出是输入信号和电路状态的函数，更严格地说是Mealy型有限状态机；而后者的输出仅仅是电路状态的函数，更严格地说是Moore型有限状态机，其输出方程可简化为

$$\begin{cases} z_1 = g_1(w_1, \ldots, w_l) \\ \quad\vdots \\ z_k = g_k(w_1, \ldots, w_l) \end{cases} \tag{7-4}$$

7.2 时序电路的手工分析方法

为了完整地描述时序电路的逻辑功能，需要用到上一节所介绍的驱动方程、状态方程和输出方程的概念。分析一个时序逻辑电路的功能，就是要找出这3组方程，并说明该时序电路的逻辑功能和工作特性。当然这些方法只适用于小规模逻辑电路。

7.2.1 同步时序电路分析

以下通过一个示例来了解时序电路的手工分析流程。

【例7-1】 试分析图7-2所示的同步时序电路的功能。

解：分析过程如下：

(1) 写出电路的驱动方程：

$$\begin{cases} J_1 = K_1 = 1 \\ J_2 = K_2 = A \oplus Q_1 \end{cases} \tag{7-5}$$

(2) 将驱动方程代入 JK 触发器的特性方程，得到状态方程：

$$\begin{cases} Q_1^{n+1} = J_1\overline{Q_1^n} + \overline{K_1}Q_1^n = \overline{Q_1^n} \\ Q_2^{n+1} = J_2\overline{Q_2^n} + \overline{K_2}Q_2^n = A \oplus Q_1^n \oplus Q_2^n \end{cases} \tag{7-6}$$

(3) 写出输出方程：

$$Y = \overline{\overline{Q_1^{n+1} \cdot Q_2^{n+1}} \cdot \overline{\overline{A} \cdot Q_2^{n+1}}} = Q_1^{n+1}Q_2^{n+1} + A + Q_2^{n+1} = A + Q_2^{n+1} \tag{7-7}$$

至此得到了图 7-2 电路所对应的驱动方程、状态方程和输出方程。

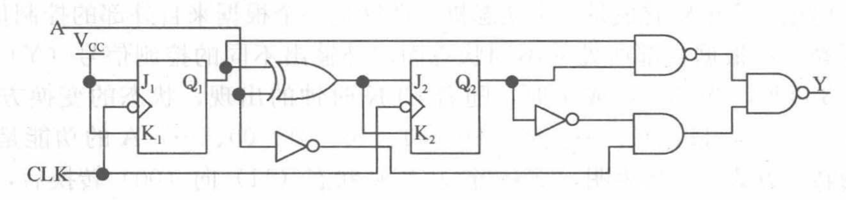


图 7-2 例 7-1 的逻辑电路图

以下通过状态转换图和时序图来描述和表达电路的逻辑功能。

状态图具有形象直观的特点，它把触发器的状态转换关系及转换条件用图形表示出来。具体做法是，首先列出状态转换表，即将输入信号和现态的所有组合状态作为输入（在本例中是 A 和现态的 Q_1 和 Q_2），然后根据输出方程和状态方程，逐行填入输出 Y 的值和次态 Q^{n+1}（Q_1^{n+1}、Q_2^{n+1}）的值。由此可列出状态转换表，如表 7-1 所示。

注意表中的输出值 Y 是属于次态时刻的数值，即 $A+Q_2^{n+1}$。

表 7-1 状态转换表

输入	现 态		次 态		输出
A	Q_2^n	Q_1^n	Q_2^{n+1}	Q_1^{n+1}	Y
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	1	0	0	1	1
1	1	1	1	0	1

为了更直观地了解电路的逻辑功能，还可以根据状态转换表画出状态转换图。在状态转换图中，每一个圆圈表示电路的一个状态，圈内的数字表示状态编码。箭头表示状态转移方向，箭头旁标注的数字表示现态的输入和输出。本题的状态图如图 7-3 所示。

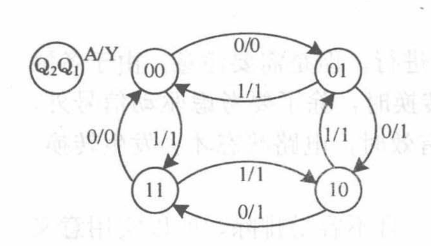


图 7-3 状态转换图

图 7-4 是此例的时序波形图。图中给出了 A 当高电平和低电平时在时钟脉冲 CLK 连续作用下，寄存器状态 Q_2、Q_1 和输出 Y 的电压波形。图 7-4 显示，每一

次状态的转换都是在 CLK 的下降沿发生的。这是由图中的 JK 触发器本身的逻辑功能决定的。

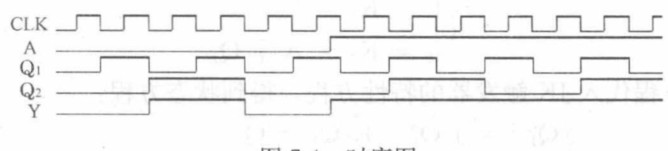


图 7-4 时序图

其实，是可以从不同的角度去理解图 7-2 电路功能的。从状态图和时序图可以看出此电路有一种计数器的功能，即这是一个加减可控两位计数器。当加减控制信号 A=0 时进行加法计数；在时钟脉冲作用下，Q_2Q_1 的值从 00 到 11 递增，每 4 个脉冲电路状态循环一次。当 A=1 时进行减法计数，在时钟脉冲作用下，Q_2Q_1 的值从 11 到 00 递减。

但也可以把这个电路看成是一个状态机。即这是一个根据来自外部的控制信号（A），在时钟的驱动下，根据内部所处的不同状态而向外输出不同的控制信号（Y）的逻辑模块。由图 7-3 可见，当 A=0 或 1 时，随着 CLK 时钟的出现，状态的变换方式分别是 Q_2Q_1=00、01、10、11、00、…，或=00、11、10、01、00、…。A 的功能是控制状态机中状态的转换方式。分析表明，唯有当 Q_2Q_1 从状态（11）向（00）转换后，或从状态（00）向（01）转换后，控制信号 Y 才输出低电平脉冲，其他情况始终保持高电平。

这个状态机的状态变量就是 Q_2Q_1，状态数是 4，即所谓有限状态。在这 4 个状态中，Q_2Q_1 所对应的从 00 到 11 的值即状态编码。状态编码可以是任意形式的，对于此 4 状态的有限状态机，其状态编码不一定总是 00、01、10、11。此外，（$Q_2^nQ_1^n$）向（$Q_2^{n+1}Q_1^{n+1}$）转换的逻辑机制是由电路中的组合电路元件组成的所谓状态译码器来决定的。

状态机有很宽的实用领域，后文将详细介绍有限状态机的工作原理和设计方法。

通过对此例的分析，可以得出手工分析时序逻辑电路的一般步骤：

（1）根据逻辑电路图写出每个触发器输入端的逻辑函数式，由此得到驱动方程。

（2）将驱动方程代入每个触发器的特性方程，得到状态方程。

（3）根据给定的逻辑图写出输出方程。

（4）根据状态方程和输出方程，列出状态转换表，画出状态转换图和时序图。

（5）根据状态转换图说明该时序电路的逻辑功能。

当然这些方法只适用于小规模逻辑电路。

7.2.2 异步时序电路的分析举例

在分析异步时序电路时，仍然可按照以上的分析步骤进行。但是需要注意，由于各触发器没有公共的时钟脉冲，所以在分析各触发器的状态转换时，除了要考虑驱动信号外，还要考虑时钟脉冲 CP 端的情况。触发器只有在 CP 脉冲有效时，电路状态才会发生转换。否则触发器状态将保持原状态不变。

异步时序电路的最大缺点是速度低，易产生冒险竞争，且不容易排除，所以实用意义不大。但尽管如此，这里介绍的内容仍有助于从不同角度去认识时序电路的特性。

【例 7-2】试分析图 7-5 所示的异步时序电路的功能。

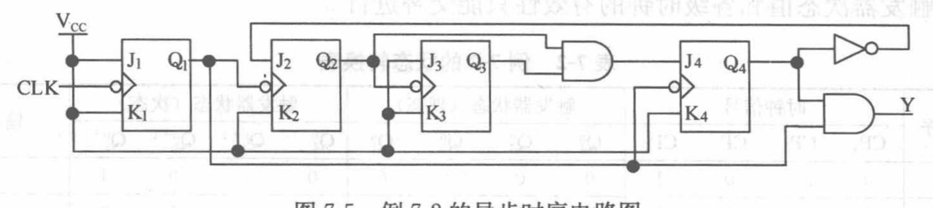


图 7-5 例 7-2 的异步时序电路图

解：(1) 根据图 7-5 写出时钟脉冲逻辑方程和驱动方程。注意时钟脉冲负跳变触发。图 7-5 中 3 个触发器的时钟脉冲逻辑方程为

$$\begin{cases} CP_1 = CLK\downarrow \\ CP_2 = CP_4 = Q_1\downarrow \\ CP_3 = Q_2\downarrow \end{cases} \Rightarrow \begin{cases} CP_1 = CLK\downarrow = (1)\rightarrow(0) \\ CP_2 = CP_4 = Q_1^n(1)\rightarrow Q_1^{n+1}(0) \\ CP_3 = Q_2^n(1)\rightarrow Q_2^{n+1}(0) \end{cases} \tag{7-8}$$

在式 (7-8) 中↓表示时钟信号的下降沿。若规定各触发器的时钟将产生有效触发边沿时 CP=1，否则 CP=0。即只有当现态为 1，次态为 0 的情况，CP=1。注意第一级触发器的时钟 CLK，总是有 1 到 0 的有效跳变，因此，恒有 $CP_1=1$。

驱动方程为

$$\begin{cases} J_1 = K_1 = 1 \\ J_2 = \overline{Q_4^n}, \quad K_2 = 1 \\ J_3 = K_3 = 1 \\ J_4 = Q_2^n Q_3^n, K_4 = 1 \end{cases} \tag{7-9}$$

(2) 将驱动方程代入各触发器的特性方程，求出各触发器的状态方程：

$$\begin{cases} Q_1^{n+1} = \overline{Q_1^n}\cdot CP_1 \\ Q_2^{n+1} = \overline{Q_4^n}\,\overline{Q_2^n}\cdot CP_2 \\ Q_3^{n+1} = \overline{Q_3^n}\cdot CP_3 \\ Q_4^{n+1} = Q_2^n Q_3^n \overline{Q_4^n}\cdot CP_4 \end{cases} \tag{7-10}$$

必须说明，式 (7-10) 只是当 CP 时钟有效情况下 (即 CP=1) 的各级触发器的次态表述。如果对应的时钟 CP 处于无效情况下，即 CP=0 时，以上各级次态并非等于 0，而是保持现态，即

$$Q_i^{n+1} = Q_i^n, \quad i = 1,2,3,4$$

(3) 根据图 7-5 得输出方程：$Y=Q_1^{n+1}Q_4^{n+1}$。注意 Y 的输出必须是次态值。

(4) 列出状态转换表，如表 7-2 所示。

对于同步逻辑，由于电路中的触发器在时钟作用下是同步变化的，即对于每一时钟的到来，状态方程总是成立的，所以可以直接利用状态方程，把状态转换表中的次态值一次性写出来。然而，对于异步逻辑却不行，这是因为异步逻辑的各级触发器的时钟的有效是有条件的，这导致其状态方程 (如式 (7-10)) 的成立也是有条件的。

于是，对于异步逻辑的状态转换表，在已知现态的情况下，次态的写出要按级逐步写

出，在这个过程中，还要根据时钟脉冲方程，逐级确定各级脉冲的有效性。这就要求，获取各级触发器次态值和各级时钟的有效性只能交替进行。

表 7-2 例 7-2 的状态转换表

时钟顺序	时钟信号				触发器状态（现态）				触发器状态（次态）				输出 Y
	CP_4	CP_3	CP_2	CP_1	Q_4^n	Q_3^n	Q_2^n	Q_1^n	Q_4^{n+1}	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	
1	0	0	0	1	0	0	0	0	0	0	0	1	0
2	1	0	1	1	0	0	0	1	0	0	1	0	0
3	0	0	0	1	0	0	1	0	0	0	1	1	0
4	1	1	1	1	0	0	1	1	0	1	0	0	0
5	0	0	0	1	0	1	0	0	0	1	0	1	0
6	1	0	1	1	0	1	0	1	0	1	1	0	0
7	0	0	0	1	0	1	1	0	0	1	1	1	0
…	…				…				…				…

这里以表 7-2 为例，说明异步逻辑状态转换表的获取步骤：

① 首先填写第一个时钟行的内容：电路的第一个现态值可以任意设，因为它总是要出现的，所以不妨设（$Q_4^nQ_3^nQ_2^nQ_1^n$）=0000。

② 逐级确定电路的四级时钟信号（假设时钟有效为 1，无效为 0）。对于第一级的时钟，由于 CP_1=CLK，即直接与 CLK 相接，所以恒有 $CP_1=1$。然后将 $\overline{Q_1^n}=1$ 和 $CP_1=1$ 代入式（7-10）得 $Q_1^{n+1}=1$。于是得到了第一个时钟行的 Q_1^{n+1} 的值，这里是 1。

③ 接下来求 CP_2。由于第一行中 $Q_1^n=0$，$Q_1^{n+1}=1$，不是下降沿，根据式（7-8）$CP_2=0$，$CP_4=0$。于是据此，式（7-10）对应的 Q_2^{n+1} 和 Q_4^{n+1} 保持原值，即照抄其现态的值 0 和 0。至此，已知第一行中的次态的值是 0x01，x 是未知，即 Q_3^{n+1} 仍未知。

④ 为求 Q_3^{n+1}，先求 CP_3。由于 $Q_2^n=0$，$Q_2^{n+1}=0$，也不是下降沿，根据式（7-8）$CP_3=0$。将 CP_3 的值代入式（7-10），即保持原态，照抄 Q_3^n 的值，得 $Q_3^{n+1}=0$。

最后获得第一行的时钟值是 0001，次态值是 0001。

⑤ 下面填写对应时钟第二行的值。在现态这一栏照抄上一行次态的值 0001；由于第一级时钟是已知的，即恒为 1，于是就很容易从式（7-10）获得次态 $Q_1^{n+1}=0$，即对现态第一位取反。接下去就是重复以上 4 个步骤，直至写满 16 行。

（5）根据状态表（表 7-2）可以画出图 7-6 所示的状态图。

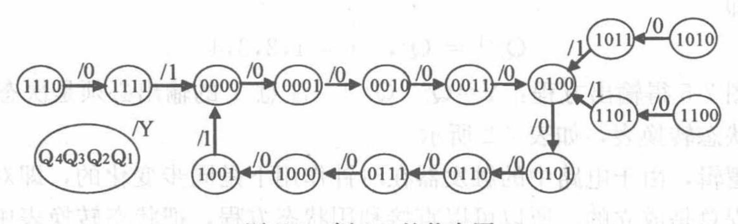


图 7-6 例 7-2 的状态图

（6）根据状态转换图可以分析和说明该时序电路的逻辑功能。

由图 7-6 可见，有效序列共 10 组，这是一个十进制异步加法计数器。无效状态没有

形成无效循环，因此是可以自启动的十进制计数器。

所谓自启动是指当计数器在上电后，其内部的触发器的状态在一开始是处于一种不可确定的随机状态，如果这种状态正好处于非法状态（无效状态），即处于正常工作状态序列以外的状态，如例 7-2 电路的状态 1110（假如此电路作为一个十进制加法计数器），而且此状态正好处于自循环状态，那么，这个计数器就一直无法进入正常工作流程，这种情况称为无法自启动。

图 7-7 是此电路的时序仿真波形图，该图清晰显示了一个十进制加法计数器的工作波形。当此计数器加到 9（1001）时，Y 作为进位，输出高电平。

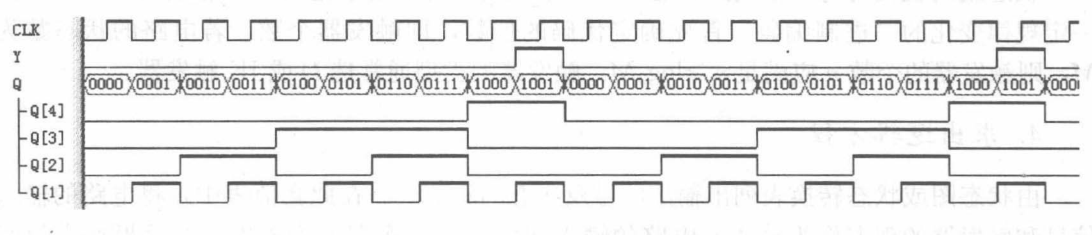


图 7-7 例 7-2 的时序仿真波形图

注意：如果利用 Quartus Ⅱ 对图 7-5 的电路进行仿真，必须对每个 JK 触发器的时钟端加一个反相器。这是因为 Quartus Ⅱ 库中的 JK 触发器都是上升沿触发的。

7.3 时序电路的手工设计方法

时序逻辑电路的手工设计实际上是以上手工分析的逆过程，以下将给出时序逻辑电路的设计方法。即根据给定的逻辑功能要求，设计一组相应的驱动方程、状态方程和输出方程，给出符合逻辑要求的时序逻辑电路，并画出与之对应的逻辑图。

7.3.1 时序电路的手工设计步骤

从传统的低速小规模数字电路设计的基本要求来看，采用手工设计方法的时序逻辑电路的设计，要求采用尽量少的触发器和逻辑门来实现所需的逻辑功能。

基于手工设计方法的小规模时序逻辑电路设计的主要步骤如下。

1. 建立原始状态图

按以下步骤完成分析，建立原始状态图：

（1）确定电路模型。分析电路的输入条件和输出要求，确定输入变量、输出变量和电路应有的状态数。

（2）定义输入、输出状态和每个状态的含义，并对各状态按一定的规律编号。

（3）按设计要求画出电路的状态转换图和状态转换表。

2. 状态化简

为了使所设计的电路使用尽量少的元件，必须对原始状态图进行化简，消除多余的状态，保留有效状态。检查电路中是否存在等价状态，如果存在等价状态，则将其合并。所谓等价状态，是指如果存在两个或两个以上电路状态，在相同的输入条件下不仅有相同的输出，而且转向同一个次态，则称这些电路状态为等价状态。

3. 状态编码

状态编码就是为每一个电路状态确定一个代码。为了便于记忆，状态编码一般选用按一定规律变化的二进制编码。首先确定代码的位数，即触发器个数。若电路的状态数为 M，则触发器的个数 n 应满足 $n \geqslant \log_2 M$。触发器的类型通常选 D 或 JK 触发器。

4. 求出逻辑方程

由状态图或状态转换表列出输出信号及次态的真值表。在此真值表中，将电路的输入信号和触发器的现态作为输入；电路的输出和触发器的次态作为输出。然后根据真值表（或直接由状态转换表）画出相应的卡诺图。最后求出电路的输出方程和状态方程，并根据所选触发器的类型和对应的特性方程，求出各触发器的驱动方程。

5. 画出电路图并检查电路的功能

根据状态方程、驱动方程和输出方程画出逻辑电路图。检查电路是否具有自启动能力，就是将无效状态代入状态方程依次计算次态，检验电路是否能够进入有效循环，如果不能，则应对设计进行修改。这有两种解决方法，一种是通过预置的方法，在开始工作时将电路预置成某一有效状态；另一种是修改设计，使电路能够自启动。

7.3.2 设计举例

以下通过一个设计实例来具体说明同步时序逻辑电路的设计流程。

【例 7-3】 要求使用 D 触发器设计一个同步 8421 BCD 码的十进制加法计数器。

解：(1) 根据设计要求，该电路没有输入变量（除时钟信号外），有一个输出变量 Y 表示进位信号。可直接得到原始状态图如图 7-8 所示。

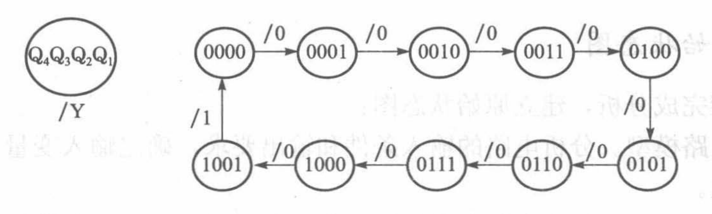
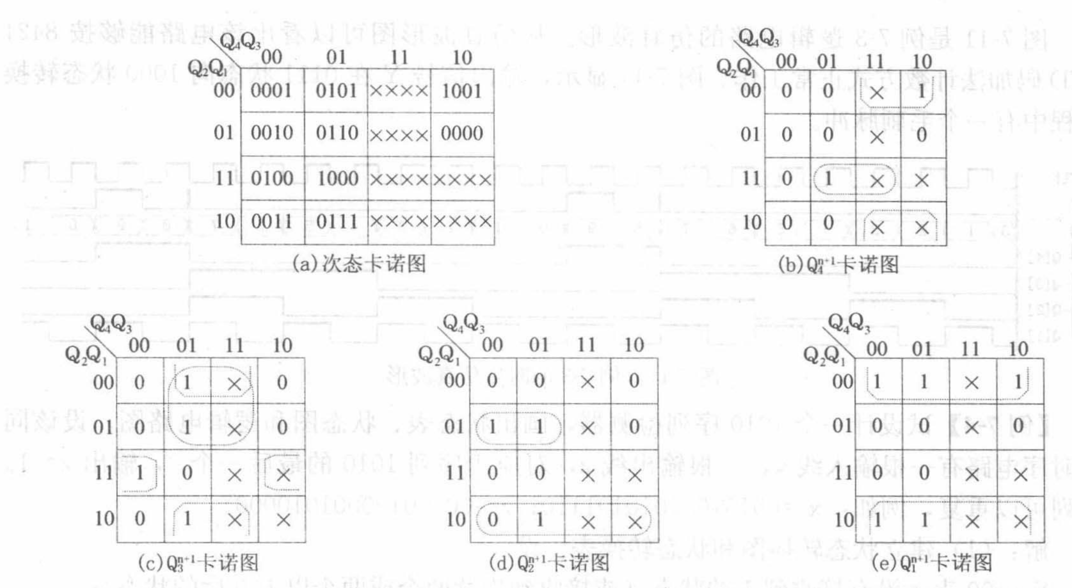


图 7-8 例 7-3 的状态图

(2) 由此状态图可得输出方程 $Y = Q_4 Q_1$，以及次态卡诺图如图 7-9 所示。



Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	0001	0101	××××	1001
01	0010	0110	××××	0000
11	0100	1000	××××	××××
10	0011	0111	××××	××××

(a)次态卡诺图

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	0	0	×	1
01	0	0	×	0
11	0	1	×	×
10	0	0	×	×

(b)Q_4^{n+1}卡诺图

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	0	1	×	0
01	0	1	×	0
11	1	0	×	×
10	0	1	×	×

(c)Q_3^{n+1}卡诺图

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	0	0	×	0
01	1	1	×	0
11	0	0	×	×
10	0	1	×	×

(d)Q_2^{n+1}卡诺图

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	1	1	×	1
01	0	0	×	0
11	0	0	×	×
10	1	1	×	×

(e)Q_1^{n+1}卡诺图

图 7-9 例 7-3 的次态卡诺图

（3）将图 7-9（a）拆分出对应 4 个次态变量 $Q_4^{n+1}Q_3^{n+1}Q_2^{n+1}Q_1^{n+1}$ 的卡诺图（b）、（c）、（d）、（e），然后由它们可求得各触发器的状态方程为

$$Q_4^{n+1} = Q_3Q_2Q_1 + \overline{Q}_1Q_4$$

$$Q_3^{n+1} = \overline{Q}_3Q_2Q_1 + Q_3\overline{Q}_2 + Q_3\overline{Q}_1 = \overline{Q}_3Q_2Q_1 + Q_3\,\overline{Q_2Q_1}$$

$$Q_2^{n+1} = \overline{Q}_4Q_1\overline{Q}_2 + \overline{Q}_1Q_2$$

$$Q_1^{n+1} = \overline{Q}_1$$

（4）D 触发器的特性方程是 $Q^{n+1}=D$，因此可直接得到各触发器的驱动方程为

$$D_4 = Q_3Q_2Q_1 + \overline{Q}_1Q_4$$

$$D_3 = \overline{Q}_3Q_2Q_1 + Q_3\overline{Q}_2 + Q_3\overline{Q}_1 = \overline{Q}_3Q_2Q_1 + Q_3\,\overline{Q_2Q_1}$$

$$D_2 = \overline{Q}_4Q_1\overline{Q}_2 + \overline{Q}_1Q_2$$

$$D_1 = \overline{Q}_1$$

（5）根据驱动方程和输出方程画出逻辑电路图，如图 7-10 所示。

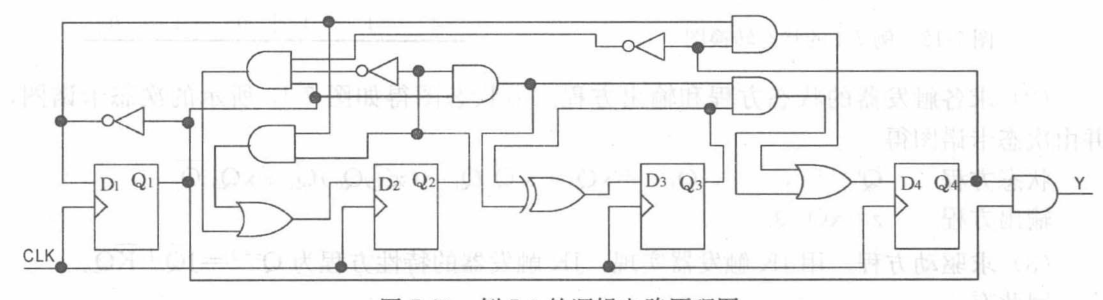


图 7-10 例 7-3 的逻辑电路原理图

最后还要验证电路的自启动性能。可以将无效状态 1010～1111 分别代入状态方程进行计算，可以验证在 CLK 脉冲作用下，都能回到有效状态，因此该电路能自启动。

图7-11是例7-3逻辑电路的仿真波形。从仿真波形图可以看出该电路能够按8421 BCD码加法计数方式正常工作。图7-11显示，输出信号Y在0111状态向1000状态转换过程中有一个毛刺脉冲。

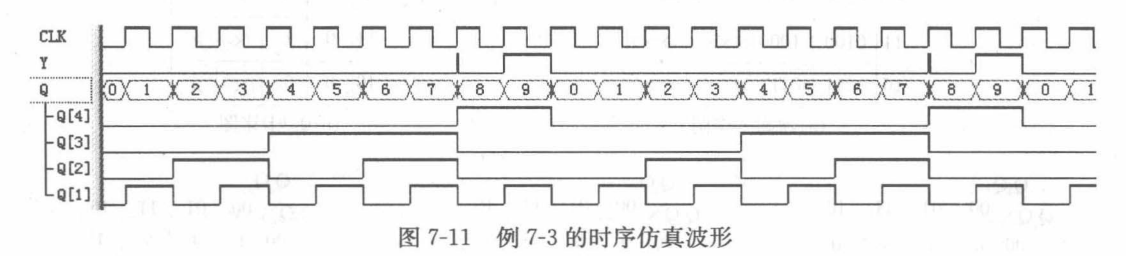


图7-11　例7-3的时序仿真波形

【例7-4】 试设计一个1010序列检测器，画出状态表、状态图和逻辑电路图。设该同步时序电路有一根输入线x，一根输出线z，对应于序列1010的最后一个0，输出z=1。序列可以重复，例如：x=00101001010101110；z=00000100001010000。

解：(1) 建立状态转换图和状态转换表。设：

S_0=00 表示没有接收到1的状态（或接收到连续两个或两个以上0后的状态）；

S_1=01 表示接收到一个1（或连续多个1）以后的状态；

S_2=10 表示接收到10以后的状态；

S_3=11 表示接收到101以后的状态。

根据题意得状态转换图，如图7-12所示，由状态转换图得状态转换表，如表7-3所示。

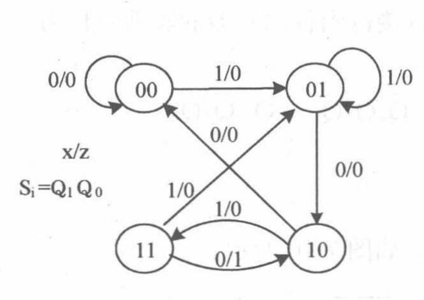


图7-12　例7-4的状态转换图

表7-3　例7-4的状态转换表

x	Q_1^n	Q_0^n	Q_1^{n+1}	Q_0^{n+1}	z
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0

(2) 求各触发器的状态方程和输出方程。由状态图得如图7-13所示的次态卡诺图，并由次态卡诺图得

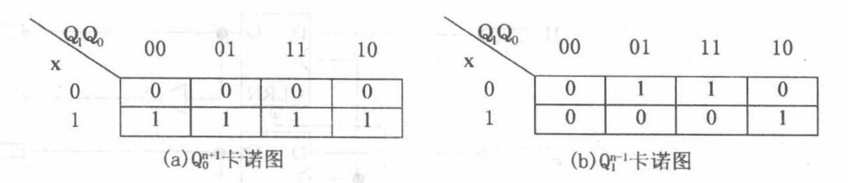
状态方程　$Q_0^{n+1}=x$　　$Q_1^{n+1}=\bar{x}Q_0+x\,\overline{Q_0}\,Q_1=(x\oplus Q_0)Q_1+\bar{x}Q_0\,\overline{Q_1}$

输出方程　$z=\bar{x}Q_0Q_1$

(3) 求驱动方程。用JK触发器实现，JK触发器的特性方程为$Q^{n+1}=J\overline{Q}+\overline{K}Q$。

因此有

$$\begin{cases}J_0=x & K_0=\bar{x}\\ J_1=\bar{x}Q_0 & K_1=\overline{x\oplus Q_0}\end{cases}$$



x \ Q_1Q_0	00	01	11	10
0	0	0	0	0
1	1	1	1	1

x \ Q_1Q_0	00	01	11	10
0	0	1	1	0
1	0	0	0	1

图 7-13 例 7-4 的次态卡诺图

(4) 根据驱动方程和输出方程画出的逻辑电路图如图 7-14 所示。

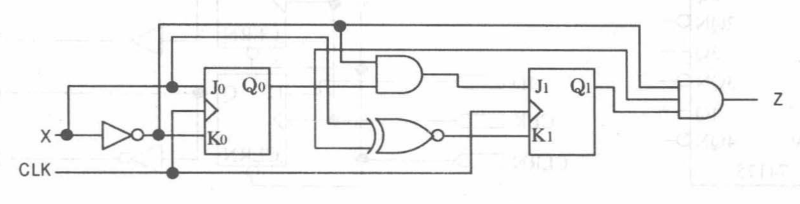


图 7-14 例 7-4 的逻辑电路原理图

7.4 寄 存 器

在时序逻辑电路中，可以说最常用的时序逻辑模块就是寄存器。在 6.6.2 节中就曾介绍过一个由 4 个 D 触发器构成的 4 位寄存器的设计和应用。以下通过介绍几种 74LS 系列通用集成寄存器器件的结构及功能特性来说明这些器件的使用方法。

为了能与以后章节的内容有更好的衔接，本节及以下类似的章节中，涉及 74 系列器件的示例都用 Quartus Ⅱ元件库中的模块符号进行举例。这些模块符号与实际 74LS 系列等通用集成器件有很好的对应关系，而且可以通过使用帮助文件 Macrofunctions 直接查阅对应型号的逻辑器件的真值表。

7.4.1 并行寄存器

用来存放二进制数据或代码的逻辑电路称为寄存器。寄存器是由具有存储功能的触发器构成的。一个触发器可以存储 1 位二进制代码；存放 *n* 位二进制代码的寄存器，则需用 *n* 个触发器来构成，称为 *n* 位并行寄存器，或 *n* 位寄存器。寄存器可以用电平触发的锁存器或是用边沿触发的触发器来构成。

图 7-15 是用 D 触发器构成的 4 位寄存器 74LS175。74LS175 的真值表如表 7-4 所示。此真值表显示，当时钟脉冲 CLK 的上升沿到来的瞬间，输入的 4 位数据 $D_0D_1D_2D_3$ 便被存入 4 个触发器中。而在 CLK 上升沿以外的其他时间，寄存器的内容保持不变，直到下一个 CLK 上升沿到来。此外，在电路中还设置了异步清 0 端 CLRN，低电平有效。当 CLRN=0 时，所有的触发器立刻被清 0。

74LS75 是用电平触发型触发器构成的 4 位寄存器。与边沿触发方式不同之处在于，电平触发的寄存器在 CLK 为高电平期间，输出状态一直随输入状态变化而变化。在 CLK 转为低电平以后，触发器的输出一直保持 CLK 从高电平回到低电平前瞬间输入端的数据。

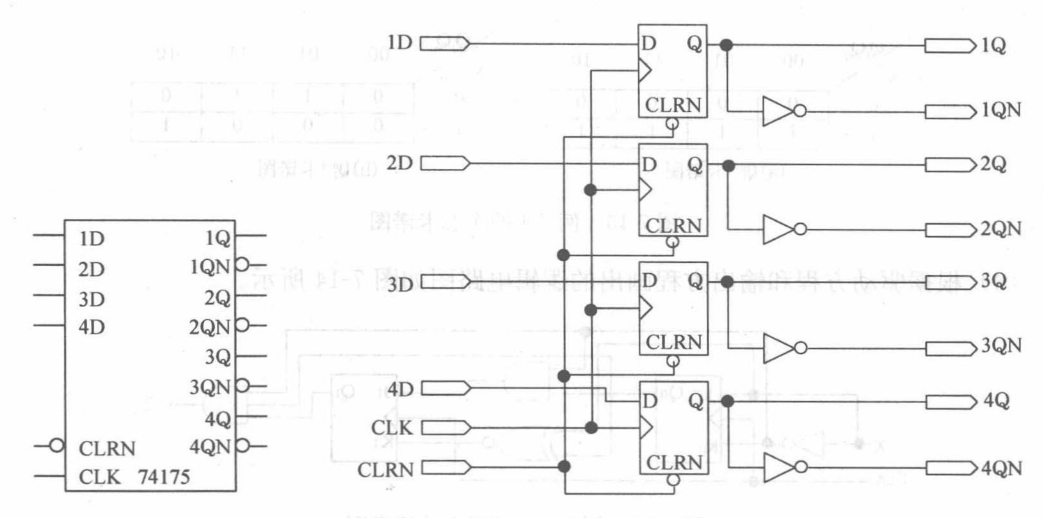


图 7-15 4 位边沿触发型寄存器 74LS175 的逻辑符号和内部逻辑电路图

表 7-4 74LS175 真值表

清 0	时钟脉冲	输 入				输 出				工作模式
CLRN	CLK	1D	2D	3D	4D	1Q	2Q	3Q	4Q	
0	×	×	×	×	×	0	0	0	0	异步清 0
1	↑	D_0	D_1	D_2	D_3	D_0	D_1	D_2	D_3	数据寄存
1	1	×	×	×	×		保	持		数据保持
1	0	×	×	×	×		保	持		数据保持

此类寄存器也常称为锁存器，其中的D触发器的功能与5.3.1节和5.3.2节介绍的触发器相同，详细的功能与结构可参考这两节的内容。

7.4.2 移位寄存器

移位寄存器不仅具有存储功能，而且在时钟脉冲信号作用下，在寄存器中存储的数据可以根据控制信号逐位左移或右移。数据可以并行输入、并行输出，或串行输入、串行输出，或并行输入、串行输出，或串行输入、并行输出，使用上灵活多变。

1. 串行输入/串行输出/并行输出移位寄存器

8位串行输入/串行输出/并行输出移位寄存器 74LS164 的逻辑符号和内部结构如图7-16所示。图中A、B是串行数据输入端，QA～QH是8位并行数据输出端，CLRN是寄存器的清0端（低电平有效），CLR是时钟脉冲输入端。

此器件的工作原理是，串行数据输入前，通过CLRN将所有寄存器清0；移位操作时，CLRN应为高电平（无效状态）。若设在8个并行数据输出端中QA是最低位，则QH是最高位。数据从A、B端（两端可以连在一起，或一端置高电平）输入，送到第一个D触发器的数据输入端D，在同步时钟脉冲CLK的作用下，第一位数据被送到QA。

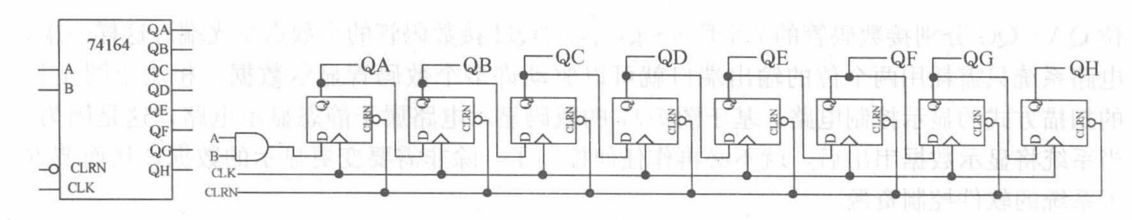


图 7-16 74LS164 的逻辑符号及其内部逻辑电路图

与此同时，QA 中原来的数据送到 QB，QB 中原来的数据被送到 QC，以此类推。这样，8 个 D 触发器中的数据被同时向右移动了一位。

当第二个时钟脉冲到来时，第二位数据从 A、B 输入，被送到 QA，8 个 D 触发器中原来的数据又被同时向右移动一位。经过 8 个同步时钟脉冲 CLK 以后，8 位数据就串行进入了移位寄存器，同时从 QA～QH 并行输出。实现了数据从串行到并行的转换。

移位寄存器 74LS164 的逻辑真值表如表 7-5 所示。

表 7-5 74LS164 的真值表

输入				输出				功能
CLK	CLRN	A	B	QA	QB	…	QH	
×	0	×	×	0	0	…	0	清 0
0	1	×	×	QA_0	QB_0	…	QH_0	保持
↑	1	1	1	1	QA^n	…	QG^n	移位
↑	1	0	×	0	QA^n	…	QG^n	移入 0
↑	1	×	0	0	QA^n	…	QG^n	移入 0

由于 74LS164 可以将串行数据变换成并行数据输出，所以常被用来扩展系统的输出端口。因为系统只需极少的端口线就可以通过多片 74LS164 的串接，将较宽位的数据并行输出。图 7-17 所示电路中，用 3 片 74LS164 将系统中的 2 位数据输出口（一位作串行数据输出 Dsin，另一位作同步时钟控制 CLK）扩展为 24 位并行数据输出口，从而大大节省了端口硬件资源。当然，这种扩展方式的代价是降低了数据输出的速度。

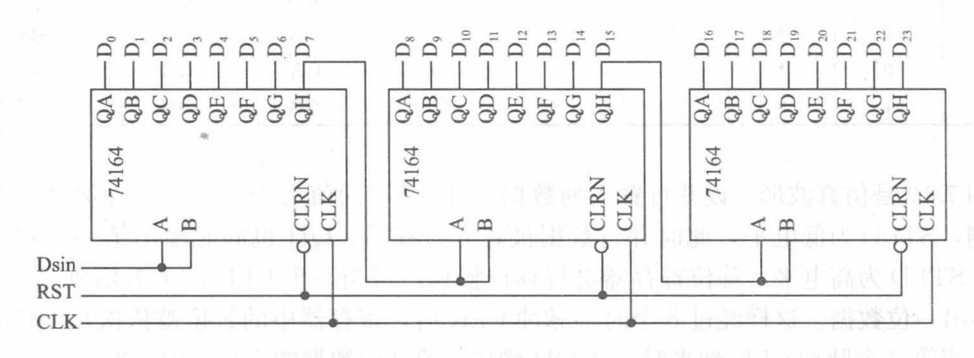


图 7-17 3 片 74LS164 将 2 位数据输出口扩展为 24 位并行数据输出口

对于一些数据输出速度要求不高的情况，图 7-17 仍有实用价值，例如用此电路构成多个数码管的显示驱动电路。即每一片 74LS164 的 8 位输出接一个 7 段数码管，其中低 7

位 QA～QG 分别接数码管的 7 个段 a～g，高位 QH 接数码管的小数点发光端。这样一来，电路系统只需耗用两个位的输出端口就可以驱动许多个数码管显示数据。相比于图 4-19 的扫描方式的显示控制电路，基于图 7-17 的数码显示电路属于静态显示电路，这是因为，当系统将显示数据串出后，就不必再作任何控制了，除非需要变更显示的数据，从而节省了系统的软件控制资源。

2. 并行输入/串行输出移位寄存器

8 位并行输入/串行输出移位寄存器 74LS165 的功能恰好与 74LS164 相反，其逻辑符号如图 7-18 所示，引脚功能分别是：CLK 是时钟输入端（上升沿有效）；CLKIH 是时钟禁止端；A～H 是并行数据输入端；SER 是串行数据输入端；QH 是输出端；QHN 是互补输出端；STLD 是移位控制/置入控制端（低电平有效）。其真值表如表 7-6 所示。74LS165 可用于扩展 8 位并行输入口。

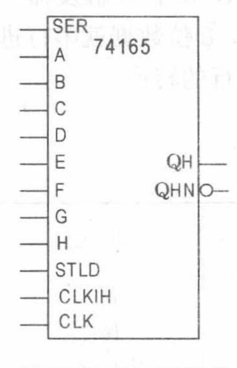


图 7-18 74LS165 的逻辑符号

74LS165 的工作原理是，当移位/置入控制端 STLD 为低电平时，并行数据（A～H）被置入寄存器，而与时钟输入（CLK）、时钟禁止（CLKIH）及串行数据（SER）均无关。当 STLD 为高电平时，并行置数功能被禁止。当时钟禁止端 CLKIH 为低电平时，允许时钟 CLK 输入，在时钟信号的作用下，寄存器中的数据进行移位操作。SER 是串行数据输入端，可通过级联，用于扩展多个移位寄存器 74LS165。

表 7-6 74LS165 真值表

输入					内部输出		输出	功能
STLD	CLKIH	CLK	SER	并行				
				A…H	QA	QB	QH	
0	×	×	×	a…h	a	b	h	置入数据
1	0	0	×	×	QA_0	QB_0	QH_0	保持
1	0	↑	1	×	1	QA_0	QG_0	移位
1	0	↑	0	×	0	QA_0	QG_0	移位
1	1	×	×	×	QA_0	QB_0	QH_0	禁止移位

图 7-19 是仿真波形。设并行置入的数据（H～A）的值为 10001011，在第 2 个 CLK 的时刻，STLD 为低电平，此时并行数据被置入寄存器，QH 也同时输出第一位数据 1。此后，STLD 为高电平，移位寄存器进行移位操作，每当一个 CLK 的上升沿到来时，从 QH 移出一位数据。这样经过 8 个时钟脉冲 CLK 后，寄存器中的数据被依次从 QH 端口送出。当第 9 个脉冲 CLK 到来时，从 QH 输出的第 9 位数据的 1 是由串行输入端 SER 送进来的数据，于是输出有：100010111。

实用中，系统可以利用很少的端口资源（仅两位）控制多片 74LS165，将外部宽位数据并行读入系统内部进行处理。

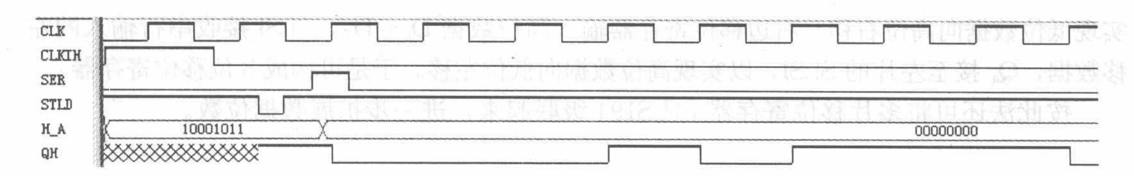


图 7-19 74LS165 的仿真波形

3. 双向移位寄存器及其应用举例

74LS194 是具有串行、并行输入，串行、并行输出及双向移位功能的寄存器，其逻辑符号如图 7-20（a）所示，其引脚：SLSI 和 SRSI 分别是左移和右移串行数据输入端；A、B、C、D 是并行数据输入端，QA 和 QD 分别是左移和右移串行输出端，QA、QB、QC 和 QD 为并行输出端。74LS194 的真值表如表 7-7 所示。

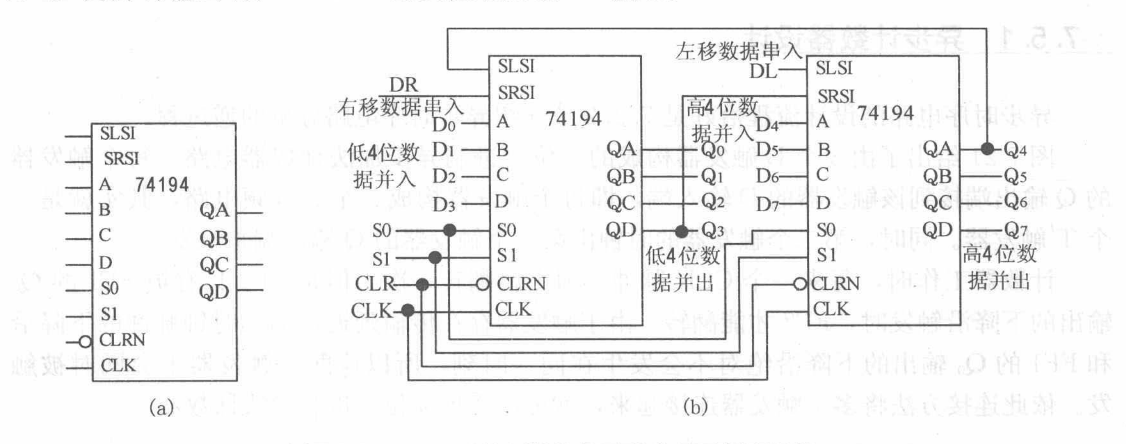


图 7-20 74LS194 的逻辑符号及其应用示例电路图

表 7-7 移位寄存器 74LS194 的真值表

输入										输出				工作模式
清 0	控制		串行输入		时钟	并行输入								
CLRN	S1	S0	SLSI	SRSI	CLK	A	B	C	D	QA	QB	QC	QD	
0	×	×	×	×	×	×	×	×	×	0	0	0	0	异步清 0
1	0	0	×	×	×	×	×	×	×	QA	QB	QC	QD	保持
1	0	1	×	1	↑	×	×	×	×	1	QA	QB	QC	右移，SRSI 串行输入，QA 串行输出
1	0	1	×	0	↑	×	×	×	×	0	QA	QB	QC	
1	1	0	1	×	↑	×	×	×	×	QB	QC	QD	1	左移，SLSI 串行输入，QD 串行输出
1	1	0	0	×	↑	×	×	×	×	QB	QC	QD	0	
1	1	1	×	×	↑	A	B	C	D	A	B	C	D	并行置数

可以将多片双向移位寄存器 74LS194 级联，扩展移位寄存器的数据位数。图 7-20（b）是利用双向移位寄存器 74LS194 扩展成一个 8 位移位寄存器的应用示例电路图。图中，左片移位寄存器输入低位数据 $D_0 \sim D_3$，其 SRSI 接收串行输入的右移数据，Q_3 接至右片的 SRSI，

实现低位数据向高位右移。右边移位寄存器输入高位数据 $D_4 \sim D_7$，SLSI 接收串行输入的左移数据，Q_4 接至左片的 SLSI，以实现高位数据向低位左移。于是可构成 8 位移位寄存器。

按此法还可将多片移位寄存器 74LS194 级联起来，进一步扩展数据位数。

7.5　计数器及其手工设计技术

数字计数器是各类数字系统中最常见的时序电路。除其基本功能外，还可以实现分频、定时、脉冲序列的产生、存储器地址信号的产生等功能。计数器有不同的种类，按其工作方式分类，有同步计数器和异步计数器；按进位体制分类，可分为二进制计数器和非二进制计数器；按计数增减方式不同，又可分为加法计数器、减法计数器和可逆计数器。实用中，特别是现代数字系统中，同步计数器更为常用，这将是本节的重点。

7.5.1　异步计数器设计

异步时序电路的设计流程恰好是 7.2.2 节介绍异步时序电路分析的逆过程。

图 7-21 给出了由 3 个 D 触发器构成的 3 位二进制异步加法计数器电路。每个触发器的 $\overline{Q}$ 输出端接到该触发器的 D 输入端，即每个触发器构成一个 2 分频电路，其实就是一个 T′触发器。同时，第二个触发器的时钟由第一个触发器的 $\overline{Q}$ 输出端来触发。

计数器工作时，每来一个 CLK 脉冲，FF1 就翻转一次。但是 FF2 只有被 FF1 的 Q_0 输出的下降沿触发时，FF2 才能翻转。由于触发器存在传输延迟，输入时钟脉冲的下降沿和 FF1 的 Q_0 输出的下降沿绝对不会发生在同一时刻，所以这两个触发器不会同时被触发。依此连接方法将多个触发器连接起来，就可以实现 n 位二进制加法计数器。

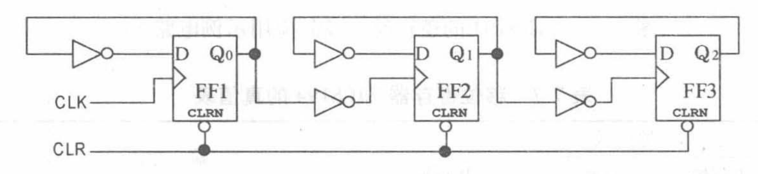


图 7-21　3 个 D 触发器构成的二进制异步加法计数器

此电路输出的时序波形如图 7-22 所示。计数开始前，通过清 0 端 CLR 将 3 个 D 触发器清 0。随着输入的每一个计数脉冲，其输出状态按二进制递增，共输出 8 个不同的状态，故被称为 3 位异步二进制加法计数器，或称为模 8 加法计数器（“模”指计数器顺序经过的状态个数，最大模是 2^n）。

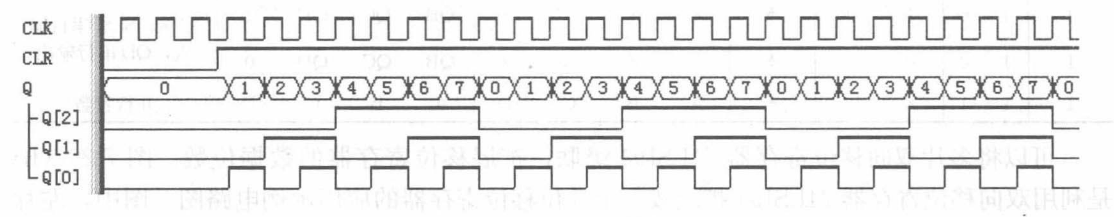


图 7-22　3 位二进制异步加法计数器仿真波形

与其他异步逻辑电路相同，异步计数器的优点是电路结构简单，缺点是它的工作速度慢。由于进位信号是逐级传递的，因此在每次输入计数脉冲以后，必须等到从最低位到最高位的进位传递完成以后，电路才能转入新的稳定状态。这种延迟情况在计数器位数多的时候更为明显。

7.5.2 同步计数器设计

为了提高计数器的工作速度，必须设法减少进位信号逐级传递的延迟时间，使所有的触发器在计数脉冲到来时同步动作，这样就必须采用同步计数器。

1. 同步二进制加法计数器设计

在同步计数器中，计数脉冲同时加到所有触发器上，而每一位触发器是否应当翻转，可以由比它低的各位的值事先判定。判断电路由组合逻辑构成，称为状态译码器。

如果将D触发器接成同步二进制加法计数器，则每一个计数脉冲到来时，最低位都应当翻转，所以 $D_0=\overline{Q_0}$，对于其他各位来说，D端的驱动方程应当为

$$D_i=(Q_0Q_1\cdots Q_{i-1})\oplus Q_i \tag{7-11}$$

对于一个用D触发器构成的同步3位二进制加法计数器可以写出其驱动方程为

$$D_0=\overline{Q_0},\quad D_1=Q_0\oplus Q_1,\quad D_2=(Q_0Q_1)\oplus Q_2 \tag{7-12}$$

还可直接写出D触发器的状态方程和电路的输出方程为

$$Q_i^{n+1}=D_i,\quad C=Q_2Q_1Q_0 \tag{7-13}$$

C为进位输出，在循环计数过程中，每当所有位都输出1时，C端输出一个高电平的进位位。根据驱动方程和输出方程可得到由D触发器构成的3位二进制加法计数器电路，如图7-23所示，其仿真波形如图7-24所示。波形图显示，该电路按二进制加法方式进行计数，当所有触发器输出高电平（计数到7）时，C端向高位输出一个进位脉冲。图7-24

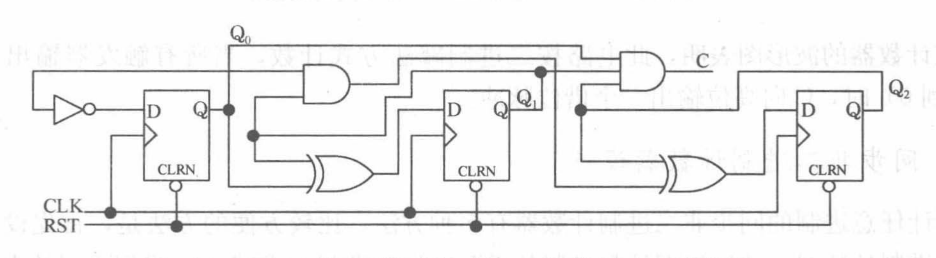


图7-23　D触发器构成的3位二进制同步加法计数器

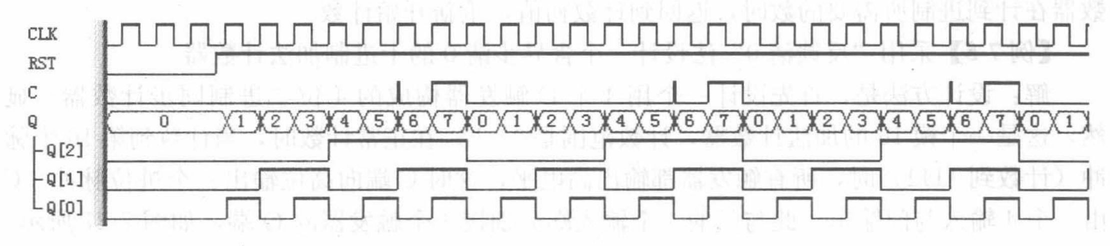


图7-24　3位二进制加法计数器的仿真波形

显示，进位信号有毛刺现象，这是因为利用 Quartus Ⅱ获得的时序仿真波形，仿真考虑了电路器件的延时情况。

用同样方法可以设计更高位数（如4位、8位等）的同步二进制加法计数器。

2. 同步二进制减法计数器设计

如果用D触发器接成同步二进制减法计数器，则每一个计数脉冲到来时，最低位都应当翻转，所以 $D_0=\overline{Q_0}$，对于其他各位来说D端的驱动方程应当为

$$D_i=(Q_0+Q_1+\cdots+Q_{i-1})\odot Q_i \tag{7-14}$$

即将原来加法器电路中的与运算改为或运算，异或运算改为同或运算。于是3位二进制减法计数器的逻辑电路图如图7-25所示，其仿真波形如图7-26所示。

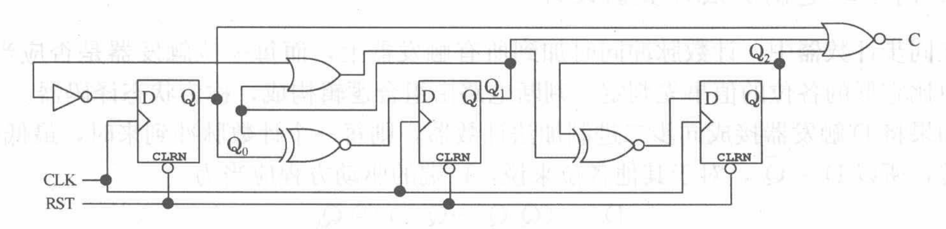


图7-25 用D触发器构成的3位二进制减法计数器电路原理图

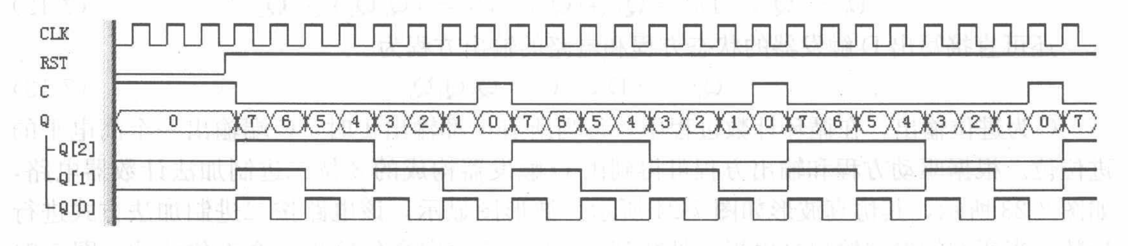


图7-26 3位二进制减法计数器的仿真波形

该计数器的波形图表明，此电路按二进制减法方式计数，当所有触发器输出低电平（计数到0）时，C向高位输出一个借位脉冲。

3. 同步非二进制计数器设计

设计任意进制的同步非二进制计数器有多种方法，比较方便的方法是，首先设计一个同步二进制计数器，然后根据计数进制的需要（如7进制，或模7），采用不同的方法使计数器在计到进制所需要的数时，返回到计数初值，重新开始计数。

【例7-5】 采用“反馈清0”法设计一个含异步清0的十进制加法计数器。

解：设计方法是，首先设计一个用4个D触发器构成的4位二进制同步计数器。显然，这是一个模16的加法计数器，计数范围是0～F。在正常计数时，当计数到第16个脉冲（计数到1111）时，所有触发器都输出高电平，这时C端向高位输出一个进位脉冲。C由一个4输入与门输出，此与门的4个输入端分别接4个触发器的Q端，如图7-27所示。但若为了实现十进制计数，可以在原来的模16加法计数器基础上增加反馈清0电路和进

位电路。模 10 加法计数器的计数输出状态如图 7-27 所示，虚线箭头指示出模 10 计数器输出状态循环路径。分析图 7-27 可知，当计数器的计数值为 1001（十进制的 9）时，进位端 C 应产生进位脉冲，该脉冲可由 Q_3Q_0 接与门产生。当计数到 1010（十进制的 10）时，应产生反馈清 0 信号使计数器清 0，该信号可由 Q_3Q_1 接与非门产生。

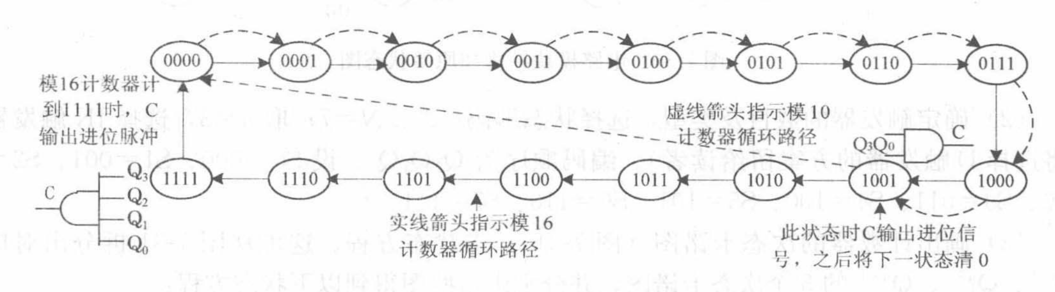


图 7-27 状态图

修改后的模 10 同步加法计数器电路如图 7-28 所示，注意图中的反馈清 0 电路和修改后的进位电路（原计数器的进位是在计数输出为 1111 时发生）。

图 7-28 电路的仿真波形如图 7-29 所示。由波形图可见，该电路实现了模 10 同步加法计数，计数范围 0～9。当计数到第 9 个脉冲（计数值 1001）时，C 向高位输出一个进位脉冲。当计数到第 10 个脉冲时，所有触发器都输出低电平，计数器回到 0，重新开始计数。

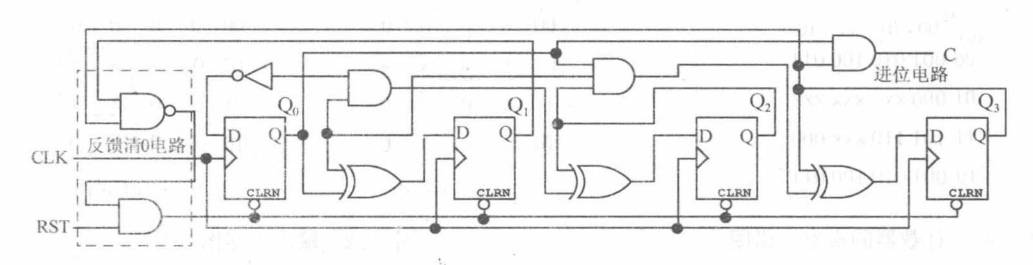


图 7-28 模 10 同步加法计数器的电路原理图

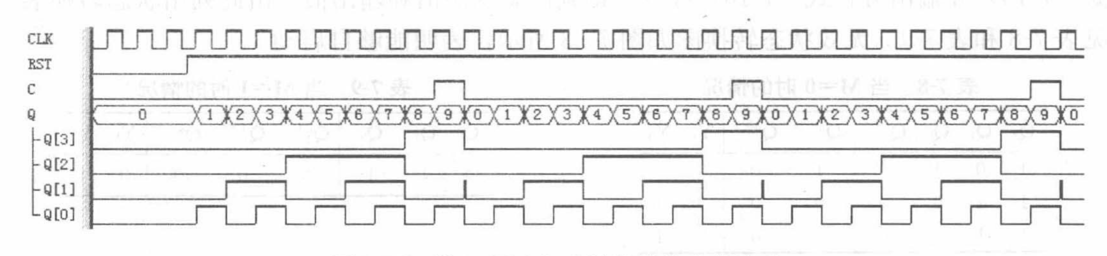


图 7-29 模 10 同步加法计数器的仿真波形

【例 7-6】 设计一个模可控同步加法计数器。要求当控制信号 M=0 时，以模 5 计数器工作；当 M=1 时，以模 7 计数器工作。

解：(1) 分析题目要求，建立原始状态图，如图 7-30 (b) 所示。

M=0 时，N=5；M=1 时，N=7。Y_0、Y_1 分别对应这两种模式输出的进位信号。

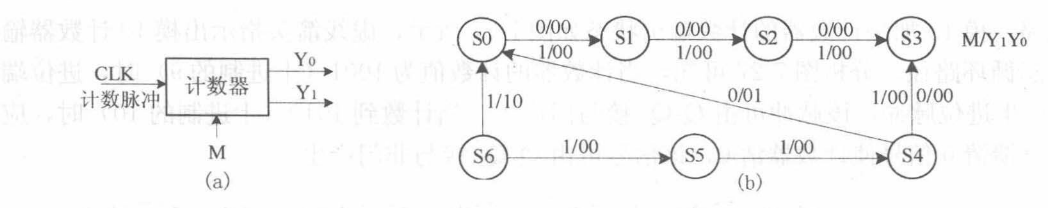


图 7-30　电路模块结构和原始状态图

(2) 确定触发器的数目及类型，选择状态编码：$2^n \geqslant N=7$；取 $n=3$，选择 JK 触发器（将选择 D 触发器的方案留给读者）。编码顺序为 $Q_2Q_1Q_0$。设 S0＝000、S1＝001、S2＝010、S3＝011、S4＝100、S5＝101、S6＝110、S7＝111。

(3) 画出计数器的次态卡诺图（图 7-31），求状态方程。这可从图 7-31 拆分出对应 Q_2^{n+1}、Q_1^{n+1}、Q_0^{n+1} 的 3 个次态卡诺图，并分别由这些图得到以下状态方程：

$$\left.\begin{aligned} Q_2^{n+1} &= M\overline{Q_1}Q_2 + Q_1Q_0\overline{Q_2} \\ Q_1^{n+1} &= \overline{Q_1}Q_0 + \overline{Q_2}Q_1\overline{Q_0} \\ Q_0^{n+1} &= M\overline{Q_1}\,\overline{Q_0} + \overline{Q_2}\,\overline{Q_0} \end{aligned}\right\} \tag{7-15}$$

画出输出卡诺图（图 7-32），求输出方程，由图 7-32 的输出卡诺图可得输出方程为

$$Y_0 = \overline{M}Q_2 \quad Y_1 = Q_1Q_2 \tag{7-16}$$

MQ_2 \ Q_1Q_0	00	01	11	10
00	001	010	100	011
01	000	xxx	xxx	xxx
11	101	110	xxx	000
10	001	010	100	011

图 7-31　计数器的次态卡诺图



(a) Y_0 的卡诺图

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	0	0	0	0
01	0	x	x	x
11	0	0	x	1
10	0	0	0	0

(b) Y_1 的卡诺图

图 7-32　输出卡诺图

(4) 检查能否自启动。将 M＝0 和 M＝1 时的无效状态值 $Q_2Q_1Q_0$ 分别代入状态方程式（7-15）和输出方程式（7-16）计算，得到次态状态值和输出值。由此列出状态转换表，见表 7-8 和表 7-9。无效状态转换图如图 7-33 所示，表明能够自启动。

表 7-8　当 M＝0 时的情况

Q_2	Q_1	Q_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	Y_1	Y_0
1	0	1	0	1	0	0	0
1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	1

表 7-9　当 M＝1 时的情况

Q_2	Q_1	Q_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	Y_1	Y_0
1	1	1	0	0	0	0	1

(5) 求驱动方程。将状态方程（7-15）与 JK 触发器的特性方程比较可得

$$J_2 = Q_1Q_0 \qquad K_2 = \overline{M\overline{Q_1}}$$

$$J_1 = Q_0 \qquad K_1 = \overline{\overline{Q_2}\,\overline{Q_0}} = Q_2 + Q_0$$

$$J_0 = M\overline{Q_1} + \overline{Q_2} \qquad K_0 = 1$$

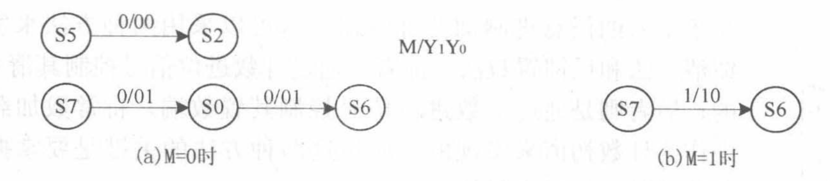


图 7-33 无效状态转换图

(6) 画逻辑图。根据驱动方程和输出方程画逻辑图，如图 7-34 所示。

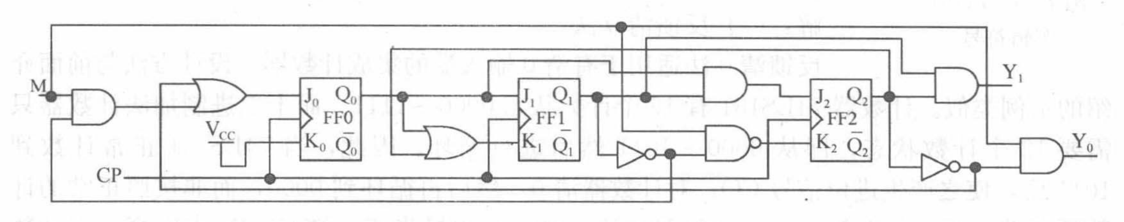


图 7-34 例 7-6 的逻辑电路图

7.6 专用集成计数器应用示例

诸如 74 系列的专用标准化集成电路器件，包括各类计数器，曾经是传统数字系统（甚至包括早期的 IBM PC 计算机系统）的重要组成部分。对于计数器等时序电路，与直接使用更小规模的触发器分离器件构成的计数器相比，专用集成器件具有更多的优势，如体积小、功能灵活、可靠性高等。专用集成计数器的种类比较多，计数进制主要以二进制和十进制为主。尽管在现代数字系统设计中，此类器件基本不会用到，但对于初学者深入了解不同类型的计数器的特点、用法和设计要点，以及对数字技术的深入学习是必不可少的。以下拟介绍几种典型的集成计数器用法。

7.6.1 用 74LS161 构成十二进制加法计数器

74LS161 是具有同步加载和异步清 0 功能的 4 位二进制加法计数器。表 7-10 是其逻辑真值表，其逻辑符号如图 7-35 所示。

表 7-10 74LS161 功能表

清 0	置 数	使 能		时 钟	预置数据				输 出				工作模式
CLRN	LDN	ENP	ENT	CLK	D	C	B	A	QD	QC	QB	QA	
0	×	×	×	×	×	×	×	×	0	0	0	0	异步清 0
1	0	×	×	↑	d3	d2	d1	d0	d3	d2	d1	d0	同步置数
1	1	0	×	×	×	×	×	×	保持				数据保持
1	1	×	0	×	×	×	×	×	保持				数据保持
1	1	1	1	↑	×	×	×	×	计数				加法计数

由表 7-10 可知，74LS161 采用异步清 0、同步置数的方式，利用这些功能可以构成

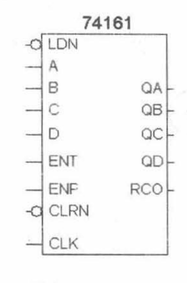


图 7-35 74161 逻辑符号

小于十六的任意进制加法计数器。这可以采用两种方法来实现，即反馈清 0 法和反馈置数法。前者是通过计数进位信号控制其清 0 端来实现的；后者则是通过计数进位信号控制其置数端，将常数加载于计数器中作为计数初值来实现的。应用这两种方法的关键是要掌握此器件属于异步清 0 和同步置数功能。

以下通过示例来说明。

【例 7-7】 用 74LS161 构成十二进制加法计数器。

解：(1) 反馈清 0 法。

反馈清 0 法适用于有清 0 输入端的集成计数器。设计方法与前面介绍的示例类似。计数器 74LS161 有 16 个计数状态 0000～1111。而十二进制加法计数器只需要 12 个计数状态，即从 0000～1011 然后进行循环。因此，当 74LS161 正常计数到 1011 后，使之产生进位信号 CO，对计数器清 0，然后再循环到 0000，而非按照正常的计数顺序进入下一个状态 1100。计数器的清 0 可以利用异步清 0 端 CLRN 来实现。当计数到 1100 时产生清 0 低电平信号从而使计数器立即清 0。当信号 CLRN 消失后，74LS161 即从 0000 进入新的计数周期。具体电路如图 7-36 所示。

由图 7-36 可见，一旦计数到 1100，电路通过一个与非门向 74161 的 CLRN 发出一个清 0 信号，禁止计数器进入 1100 状态。与例 7-5 类似，这个电路的进位信号 CO 和清 0 信号并非产自同一个状态，前者比后者要少 1，即在 1011 状态产生 CO。图 7-38 是此计数器的仿真波形。图中显示有一个毛刺脉冲，且当计数到 BH 时产生一个进位脉冲。

(2) 反馈置数法。

解法 1：反馈置数法适用于有置数端的集成计数器。利用 74LS161 构成十二进制加法计数器时，反馈置数法的优势是，可选择其 16 个计数状态 0000～1111 中的任意连续的 12 个状态作为十二进制加法计数器的计数状态，如选择 0000～1011 或 0100～1111。

若选择计数状态是 0000～1011，则计数状态的转换情况与反馈清 0 法相同，只是当计数到 1011 后，一方面输出一个进位信号；另一方面，将这个进位信号作为计数器的加载允许信号 LDN，将 0000 从并行数据入口加载到计数器中作为下一计数循环的初值。这种方案的电路如图 7-37 所示。其仿真波形也是图 7-38。

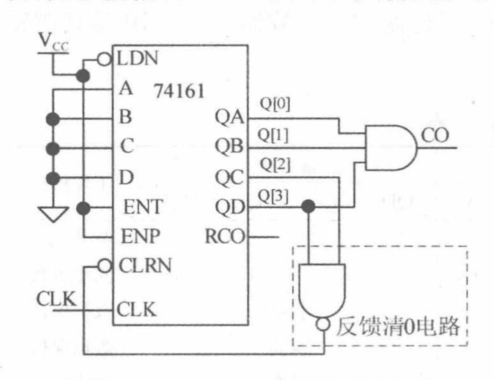


图 7-36 反馈清 0 法的计数器电路

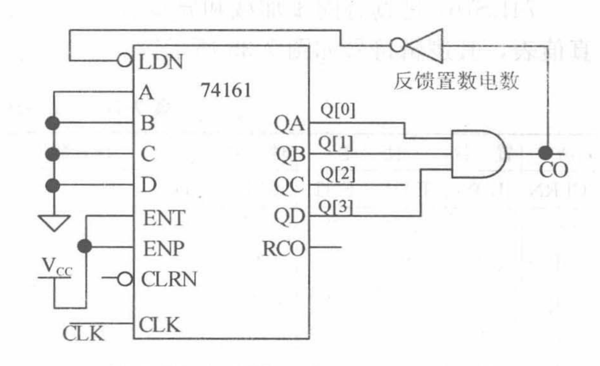


图 7-37 反馈置 0 方法的计数器电路

但与反馈清 0 法不同的是，反馈置数法生成进位信号和加载允许信号的状态是相同

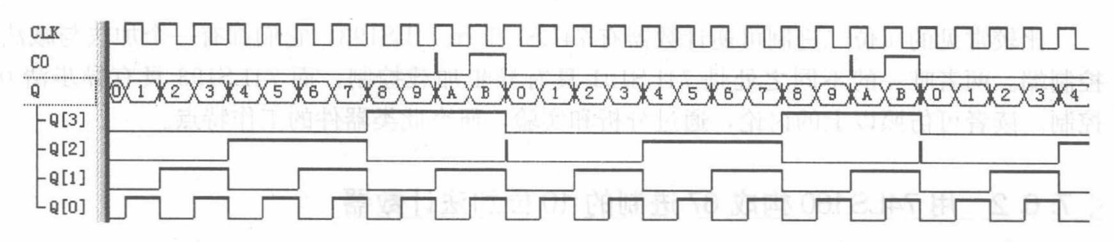


图 7-38 由 74161 构成的模 12 加法计数器仿真波形

的，都是 1011。这是因为，预置一个数进入计数器必须与时钟同步，即必须由时钟的有效边沿才能将预置数锁入计数器，因此在时间上要晚一拍。显然，这个有效时钟原本是要使计数器进入 1100 状态的。

而反馈清 0 法中控制的清 0 端 CLRN 是异步的，即不受时钟控制的。即当 CLRN 信号一出现，计数器即刻进入 0000 状态，因此可以晚一拍时钟发出清 0 操作。

解法 2：若选择计数状态是 0100～1111，即当 74LS161 正常计数到 1111 后，使其转到状态 0100。这可以通过在 74LS161 的预置数据输入端预先放置固定数据 0100，并使它的同步置数端 LDN 有效来实现。当置数信号 LDN 消失后，74LS161 即从预置数 0100 开始新的计数周期。这一方案对应的电路如图 7-39 所示；此电路的仿真波形如图 7-40 所示。注意，与波形图 7-38 一样，图 7-40 中的进位信号也有毛刺。

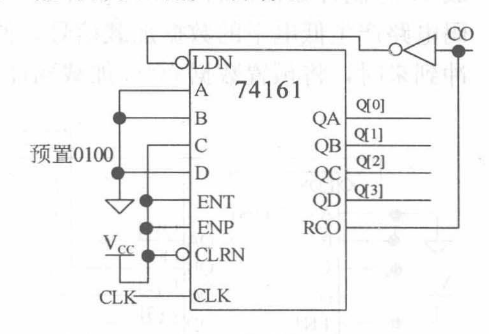


图 7-39 反馈置数法的计数器电路

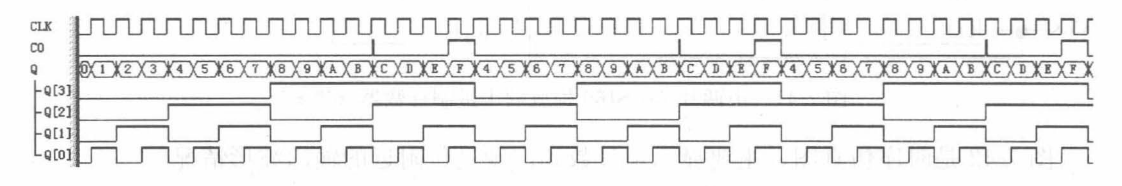


图 7-40 反馈置数 0100 构建的模 12 的加法计数器仿真波形

图 7-39 的电路显示，计数器在进入 1111 状态后，此时进位信号 RCO＝1，并行输入端的数据 0100 并没有被立即置入，而此时输出状态（1111）保持不变，直到下一个时钟脉冲的上升沿到来为止。因此，同步置数没有过渡状态，这种与时钟同步操作的情况是与异步清 0 方式不同的。

与 74LS161 功能类似的另一种型号的同步 4 位二进制计数器是 74LS163，其引脚图与 74LS161 完全相同，不同之处是 74LS163 为同步清 0 方式。

需要特别提醒的是，在工程设计中，若利用诸如图 7-36、图 7-37 或图 7-39 类似电路中的反馈置数或清 0 的方法来设计计数器，常会由于反馈信号中不确定的毛刺而影响计数器的工作可靠性（因为过宽的毛刺脉冲容易引起提前清 0 或提前置数等误操作）。所谓不确定是指毛刺的宽度、出现的位置、清 0 或加载响应时间、集成计数器的类型以及外部环境温度等因素的不确定，下一章中将对此作进一步的讨论。

比较常见的 4 位二进制可逆计数器有 74LS191 和 74LS193，它们都有一个加法与减法控制端。两者唯一的不同之处是 74LS191 具有异步加载控制，而 74LS193 具有异步清 0 控制。读者可仿照以上的讨论，通过分析和实验，研究此类器件的工作特点。

7.6.2 用 74LS160 构成 67 进制的 10 位加法计数器

与 74LS161 相比，除了 74LS160 是一个十进制加法计数器外，其余所有情况，包括清 0 和预置控制方式以及引脚端口的排列都与 74LS161 相同。

【例 7-8】 试用两片 74LS160 构成一个模为 67 的十进制加法计数器。

解：首先将两片 74LS160 串接成一个一百进制计数器，电路如图 7-41 所示，其中左侧的器件记个位数，右侧的记百位数。然后，再用反馈置数法将一百进制计数器电路改接成 67 进制计数器。若从 0 开始计数，则计数到 66 时输出进位信号 CO，并通过与非门译码电路产生低电平的数据加载信号，使两片 74LS160 的 LDN 同时为低电平，当第 67 个脉冲到来时，将预置数据 0000 加载到计数器，使计数器从 0 开始重新计数。

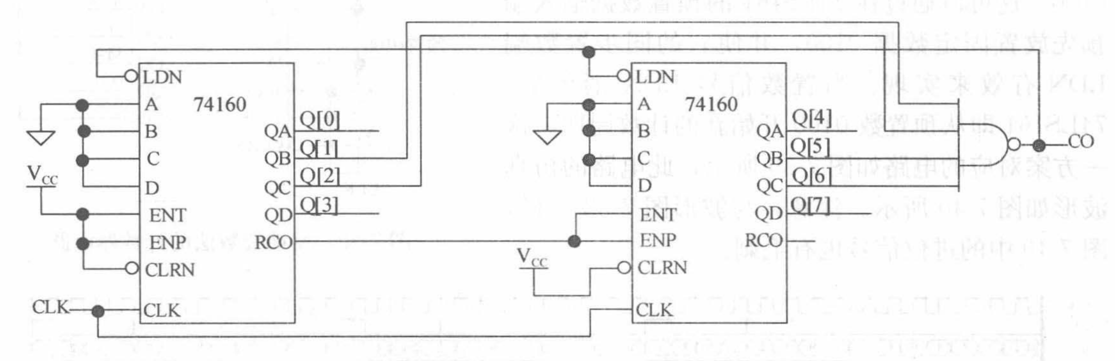


图 7-41　由两片 74LS160 构成的十进制计数器（模 67）

图 7-42 是时序仿真图，主要显示了计数到出现进位附近的输出波形情况。

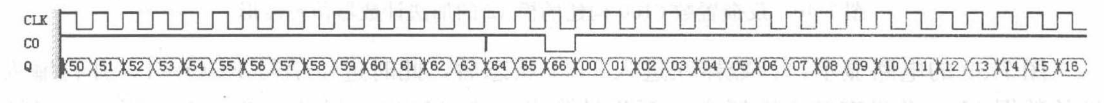


图 7-42　例 7-8 的仿真波形图

7.6.3 用 74LS161 设计一个 8 位二进制可预置计数器

可以用两片 74LS161 级联构成。采用反馈预置法可实现小于 255 的任意进制加法计数器，电路结构如图 7-43 所示。图 7-43 中，LD 是预置控制端（低电平有效），D7～D0 是 8 位二进制预置数据输入端，CLK 是时钟脉冲输入端，Q7～Q0 是计数器的 8 位二进制数据输出端，CO 是进位信号输出端。

图 7-44 是电路的仿真波形图。图中，预置数据分别为十六进制数 FA、F6 和 FA。在

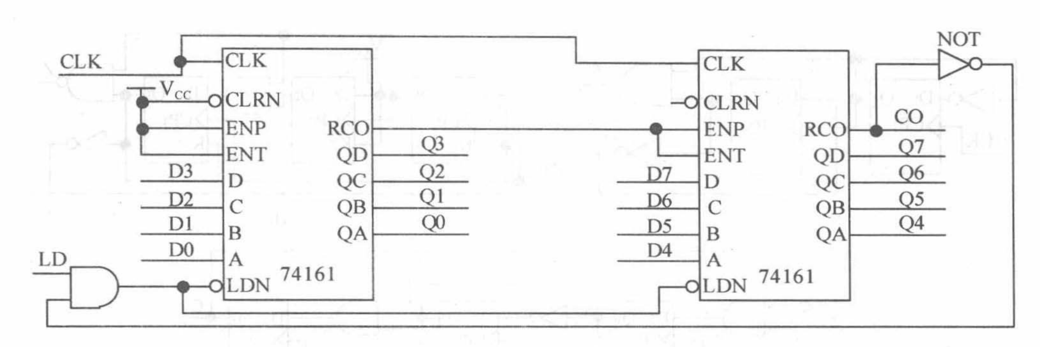


图 7-43　用 2 片 74LS161 构成 8 位二进制可预置计数器

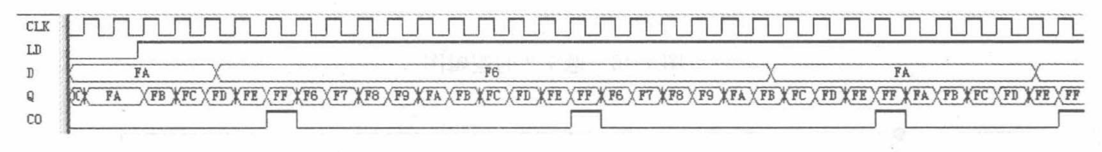


图 7-44　图 7-43 电路的仿真波形图

第 1 次计到 FFH 时，进位信号 CO 发出脉冲的同时，将预置数 F6 载入计数器。第 2 个进位脉冲也一样，直到第 3 个进位脉冲将 FA 置入计数器。波形显示，此计数器可按预置数据的设置，自动地进行循环计数。

可以把这个电路设计成一个可控的分频器，分频比可通过预置数据来控制。

习　题

7-1　试分别分析图 7-45 所示的两个同步时序逻辑电路，画出状态转换图和时序图，并说明电路的逻辑功能。

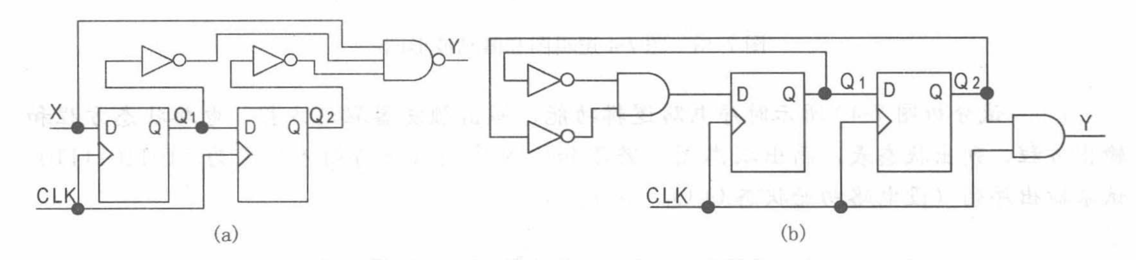


图 7-45　题 7-1 的逻辑图

7-2　试分别分析图 7-46 中的 3 个异步时序逻辑电路，画出状态转换图和时序图，并说明电路的逻辑功能。

7-3　分析图 7-47 所示电路。写出驱动方程、输出方程、状态方程、状态转换表，画出状态转换图。

7-4　画出图 7-48 电路在激励信号 CLK 和 D 的作用下，Q_1、Q_2 的输出波形。设初态为 00。

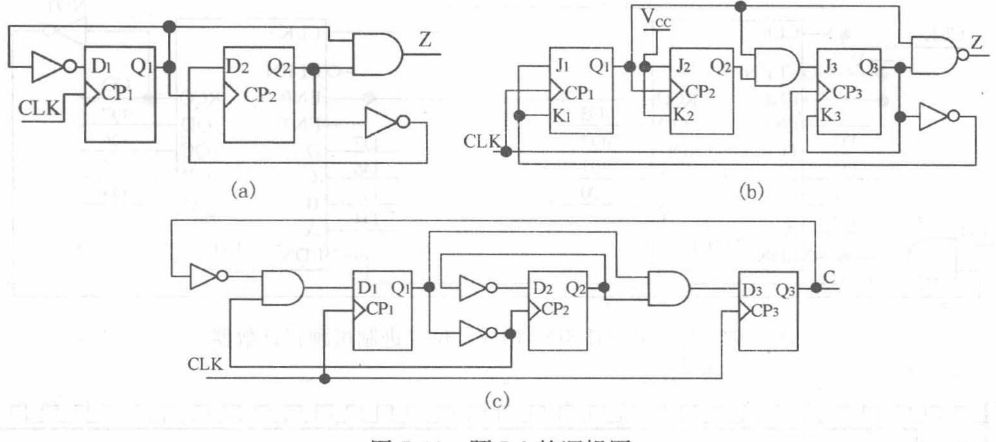


图 7-46 题 7-2 的逻辑图

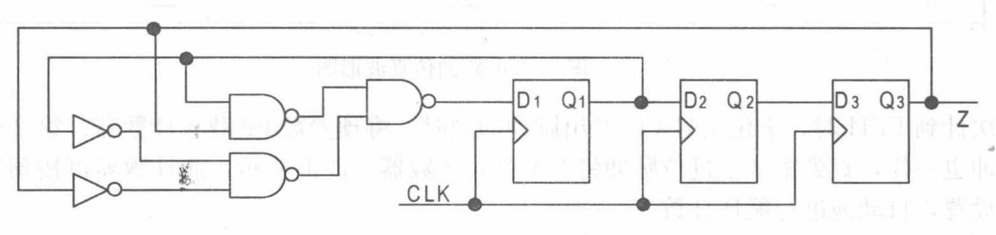


图 7-47 题 7-3 的逻辑图

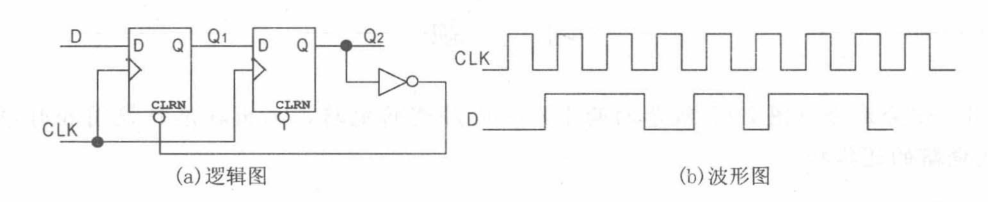


图 7-48 题 7-4 逻辑图和激励波形图

7-5 试分析图 7-49 所示时序电路逻辑功能，写出触发器驱动方程、电路状态方程和输出方程，列出状态表，画出状态图。若已知输入序列（串行输入）X 为 01011011110，试求输出序列（设电路初始状态 $Q_2Q_1=00$）。

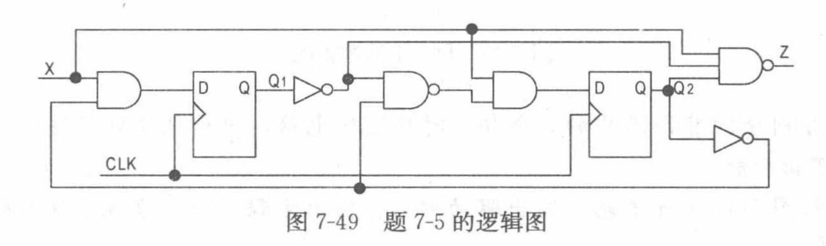


图 7-49 题 7-5 的逻辑图

7-6 分析图 7-50 所示序列检测器电路，列出状态表，画出状态图，并说明该电路具体功能。

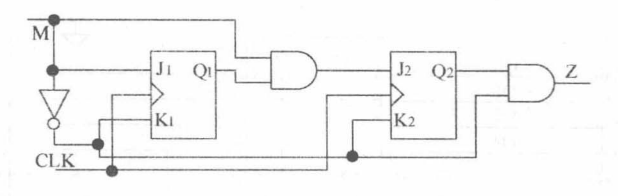


图 7-50 题 7-6 的逻辑图

7-7 试用 D 触发器设计一个 101 序列检测器，用于检测串行二进制序列，要求每当出现 101 时，检测器输出为 1，否则输出为 0，画出逻辑电路图。其典型输入输出序列如下：

输入 X：0101010001011　　　　输出 Z：0001010000010

7-8 试用 D 触发器和必要的逻辑门设计一个六进制计数器。

7-9 试设计一个电路装置以产生周期二进制序列 01011。

提示：序列 01011 计 5 位，故可设计一个模 5 计数器和译码器来实现。

7-10 设计一个巴克码信号发生器，要求自动产生周期性的 1110010 的信号序列，要求用 D 触发器和逻辑门来实现。

7-11 设计 1110 序列检测器的状态转换图，并求出最简状态转换表。

7-12 使用 JK 触发器，设计一个变模计数器。画出逻辑电路图，给出时序波形图。

要求：(1) 控制端 X=0 时，计数器的模 M=3，计数规律为

$$00 \rightarrow 01 \rightarrow 10$$

(2) 控制端 X=1 时，计数器的模 M=4，计数规律为

$$00 \rightarrow 01 \rightarrow 10 \rightarrow 11$$

7-13 设计一个同步模 7 计数器，其状态图如图 7-51 所示。用 D 或 JK 触发器和门电路设计。

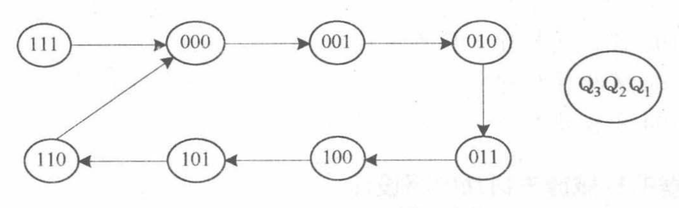


图 7-51 题 7-13 的状态图

7-14 试用 74LS194 双向移位寄存器，设计能产生序列信号 00011101 的移位寄存器型的序列信号发生器。有关 74LS194 的时序功能可以查阅相关资料，如使用帮助文件 Macrofunctions。

7-15 时序电路分析。

(1) 试分析图 7-52 所示的电路中，当 M、N 为各种不同输入时，分别完成何种功能？

(2) 如要得到模 6 计数器，应如何改接线路，M、N 需加什么信号？

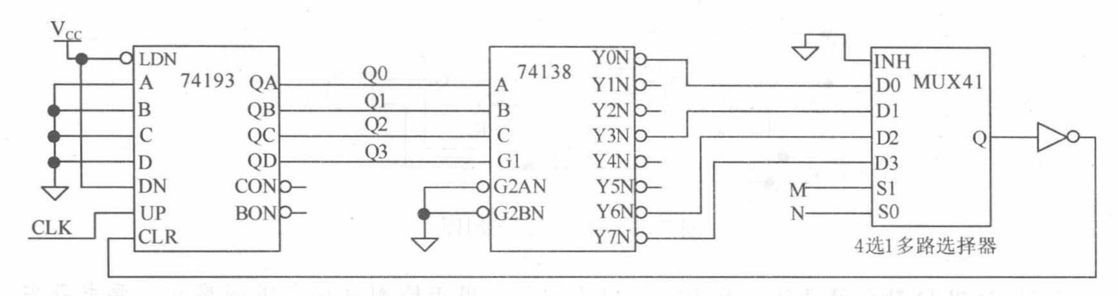


图7-52　题7-15的逻辑图



7-1　用74系列的专用集成器件设计不同类型的数字电路

实验目的：初步掌握利用Quartus Ⅱ平台对时序电路的设计、仿真及硬件实现的重要技术，熟悉在Quartus Ⅱ器件库中调用74系列宏器件的方法。

(1) 在Quartus Ⅱ平台分别验证7.6节中介绍的所有设计项目，流程包括电路编辑输入、综合、仿真（包括对仿真结果的讨论），直至下载到FPGA中进行硬件验证。具体项目包括：

① 用74161宏模块构建十二进制加法计数器。分别用反馈清0法和反馈置数法完成这项验证性设计。

② 用74160宏模块构建67进制10位加法器。

③ 用74161宏模块设计一个8位二进制可预置计数器。对于此项实验的发挥部分是，基于此计数器，设计一个可预置型分频器。即对应预置不同的8位数据，分频器将有不同的分频比。试给出此分频器预置值与分频比间的关系式。最后进行硬件验证。

(2) 此项任务主要是自主设计，流程包括电路设计、时序仿真、下载到FPGA中进行硬件验证。讨论仿真与硬件测试中出现的问题和结果。设计具体项目包括：

① 用多片74193宏模块构建8位可逆计数器。

② 用4片74164宏模块构建一个4位数码管串行静态显示电路。

③ 用2片74194宏模块构建一个8位双向移位寄存器。

7-2　基于D触发器的机械键去抖动电路设计

按照图6-40，设计一个机械键去抖动电路（注意工作频率值的确定）。首先通过仿真验证所设计的电路，然后将此设计使用到一个机械按键上。此键可以是实验系统上一个有机械抖动的键。要求按此键后，FPGA能收到一个没有任何抖动或干扰脉冲的键脉冲信号。为了证明这个去抖动电路的可行性，必须首先设计一个2位十六进制计数器，计数器的值可以通过实验系统上的两个数码管显示出来。计数器时钟端可与去抖动电路相接。于是，每按一次键后就可以观察到计数器显示值的变化。当每按一次键，如果计数器计数值只显示加1，表明去抖动电路效果良好，如果计数值大于1，表明键的抖动尚未消除。

设计要求：创建工程，绘制电路图，全程编译，对设计进行时序仿真，根据仿真波形作说明，引脚锁定编译，编程下载于 FPGA 中，在实验系统上硬件验证。最后完成实验报告。

当然也可以设计其他方法来证实去抖动的有效性。试推出新的去抖动方案，并证实之。

提示：利用两片 74161 构建 8 位二进制计数器，然后根据实验 6-5 设计一个十六进制 7 端显示译码器，并在 74161 构建的计数器的低 4 位和高 4 位输出口各接一个这样的译码器。译码器输出接 FPGA 外的两个数码管，用于验证去抖动电路的可行性。

7-3　设计一个能将信号延时 800ns 的延时电路

按照 6.6 节的设计原理和流程，设计一个 8 通道延时电路，要求能将信号延时 800ns。给出设计电路，计算工作时钟的频率，根据仿真波形作说明，编程下载于 FPGA 中，在实验系统上实现硬件验证。最后完成实验报告。

第 8 章 时序电路的自动化设计与分析

事实上，第 7 章中所介绍的时序电路的传统手工设计方法所存在的缺陷，与 6.1 节中列出的问题是相同的。例如，对于时序电路设计，传统手工技术只考虑纯逻辑的实现，至于如何优化设计结构，如何检测设计结果（如速度、可靠性等），以及如何评估电路的可行性等（如随机毛刺脉冲的发现和去除）都无从实现，因此完全无法直接适用于现代高速数字电路系统的设计。

本章首先介绍基于自动化数字设计技术的面向 74 系列宏模块逻辑电路设计技术，包括基于 Quartus Ⅱ 的时序逻辑电路的设计与分析方法，特别是一些处理问题的技巧。所给的实例向读者证明，一些表面上十分经典的数字电路常常隐含着许多传统手工数字技术无法了解和排除的性能隐患（如毛刺脉冲），使读者能更深切感受到掌握对不同功能和不同逻辑规模的实用时序逻辑电路的自动化设计技术的重要性。

然后给出数字计数器的通用设计模型，进而描述了更具一般形式的有限状态机模型，为顺利地进入现代数字技术更深入的学习铺就了良好的通途。

最后，基于状态机模型，给出了数则有实用意义的有限状态机设计示例。

8.1 用 74 系列宏模块设计数字电路

本节基于 Quartus Ⅱ 平台介绍两则使用 74 系列宏模块完成的数字电路设计的示例，读者可以由此出发，自主设计更多更好更实用的系统。在第二个实例中，进一步说明如何使用现代数字电路自动设计技术发现和处理电路中的问题。

8.1.1 用 74390 宏模块设计一个 2 位十进制计数器

图 8-1 是 2 位十进制计数器电路图（设此工程的文件名为：COUNTER10. bdf）。其核

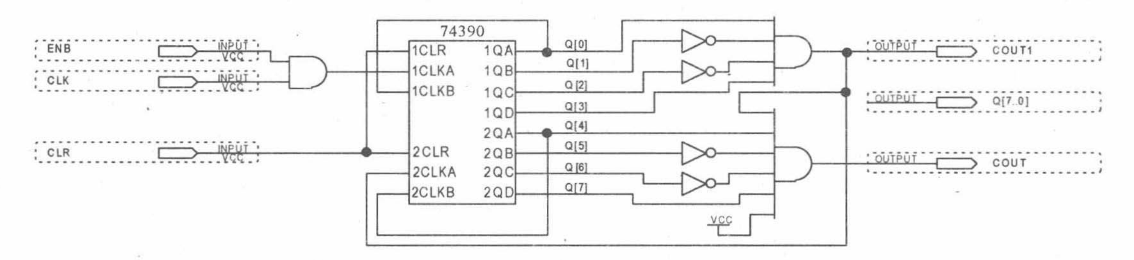


图 8-1 由 74390 构成的 2 位十进制计数器电路图，文件名 COUNTER10. bdf

心器件是74390。通过帮助文件Macrofunctions可以了解其逻辑功能。从74390的真值表（图8-2）可知，这是一个双十进制计数器。图8-1的电路构成了一个容易扩展的2位十进制计数器。输出信号COUT是最高位计数进位信号；Q[7..0]是此计数值的输出总线。鼠标双击元件74390，可以看到74390内部的结构。

74390 (Counter)

Macrofunctions

Dual Decade Counter

Default Signal Levels: GND--1CLR, 1CLKB, 2CLR, 2CLKB
VCC--1CLKA, 2CLKA

AHDL Function Prototype (port name and order also apply to Verilog HDL):

```
FUNCTION 74390 (1clr, 1clka, 1clkb, 2clr, 2clka, 2clkb)
   RETURNS (1qd, 1qc, 1qb, 1qa, 2qd, 2qc, 2qb, 2qa);
```

Inputs		Outputs			
CLR	CLK	QD	QC	QB	QA
H	X	L	L	L	L
L	˥		Count		

Possible Counting Configurations:

Decade: QA Connected to CLKB					Bi-Quinary, QD Connected to CLKA				
Count	QD	QC	QB	QA	Count	QA	QD	QC	QB
0	L	L	L	L	0	L	L	L	L
1	L	L	L	H	1	L	L	L	H
2	L	L	H	L	2	L	L	H	L
3	L	L	H	H	3	L	L	H	H
4	L	H	L	L	4	L	H	L	L
5	L	H	L	H	5	H	L	L	L
6	L	H	H	L	6	H	L	L	H
7	L	H	H	H	7	H	L	H	L
8	H	L	L	L	8	H	L	H	H
9	H	L	L	H	9	H	H	L	L

图8-2　74390的真值表

图8-1中，74390连接成两个独立的十进制计数器。CLK通过一个与门进入74390的计数器1CLKA端。与门的另一端由计数使能信号ENB控制：当ENB=1时允许计数，ENB=0时禁止计数。内部计数器1的4位输出Q[3]～Q[0]并成总线表达方式，即Q[3..0]。同时由一个4输入与门和两个反相器构成进位信号，即当计数到9（1001）时，输出进位信号COUT1。此进位信号进入第二个计数器的时钟输入端2CLKA。第二个计数器的4位计数输出是Q[7]～Q[4]。图右侧的与门与反相器一同分别构成两个计数器的进位信号。这两个计数器的总的进位信号，可由一个6输入与门和两个反相器产生，可以作为计数器扩展的进位信号。CLR是计数器的清0信号。

图8-3是此电路的仿真波形。图中，电路功能符合设计要求。CLR对系统的清0是高电平有效；ENB对CLK的计数使能也是高电平有效；低位进位信号COUT1，逢9产生进位脉冲。图8-1的电路可以作为构建更宽位计数器的基本部件。

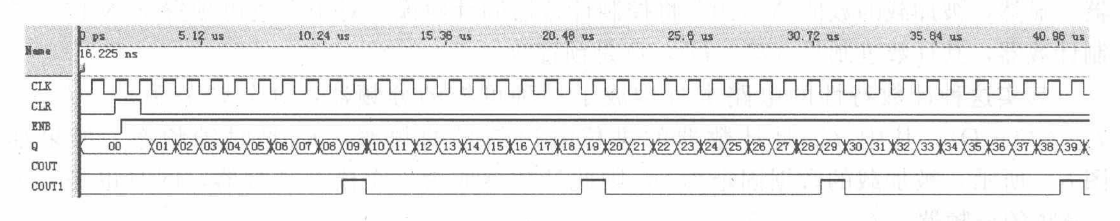


图8-3　图8-1电路的仿真波形

8.1.2 可预置型任意模计数器设计

这里以4位二进制计数器74161宏模块作为基本器件，设计一个模16以内的任意模或任意进制计数器。其方法很容易推广到更宽位计数器的应用，而且此项设计本身可以作为一个应用广泛的分频比可控分频器。

此项设计的电路图如图8-4所示。电路利用了74161的数据预置功能。图中，74161的RCO是进位输出，每当计数到F（1111）时，即输出进位脉冲。当这个进位信号引入加载端LDN时，由于LDN是同步控制的，而RCO（COUT）在出现进位脉冲期间，只要含有时钟上升沿，就能将A[3..0]的数据加载于计数器中的寄存器内。此后，计数器将

在此加载数据基础上进行加法计数。当 A[3..0] 的数据不同时，则计数的初值就不同，从而构成的计数器的进制就不同。从仿真波形图（图 8-5）可见，计数的初期，即在第一个进位脉冲出现前，74161 按普通二进制计数器的计数方式从 0 开始计到 F。此后通过加载数据输入口 D、C、B、A，把数据 A[3..0] 加载于计数器中。

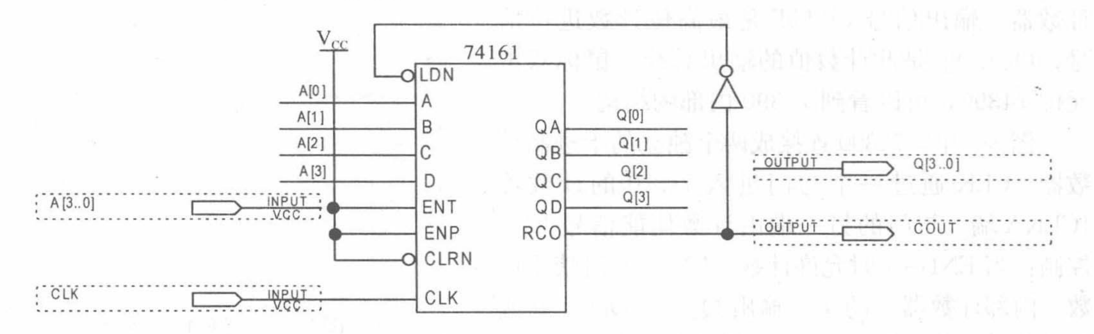


图 8-4 利用 74161 的数据预置口构成的模可控型计数器

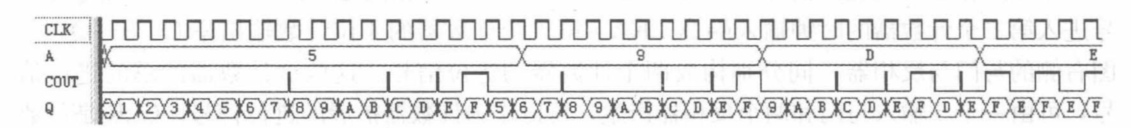


图 8-5 图 8-4 的时序仿真波形（基于 ACEX1K 系列 EP1K30TC144-3 的时序仿真）

以图 8-5 中被加载 9 后的计数情况为例，当被加载了 9 后，计数器的计数值将在 9、A、B、C、D、E、F 共 7 个计数值中循环计数，从而成为一个 16－A[3..0]＝7 进制计数器。显然，被加载的数值 A[3..0] 将控制计数器的计数模。对于此类可预置型 N 位二进制计数器，其计数进制是 (2^N-D)，D 是预置数。

其实这种计数特性的电路本身构成了一种可控的分频器，其分频关系是：$f_{cout}=f_{clk}/(2^N-D)$，其中 f_{cout} 是计数器的进位输出信号的频率，f_{clk} 是时钟频率。如果如图 8-5 所示，被加载的数据固定为 9，则此计数器成为一个模 7 计数器，同时也是一个 7 分频的分频器：$f_{cout}=f_{clk}/7$。

对于图 8-5，不难注意到图中的进位信号中有许多明显的毛刺！

尽管由于控制端是时钟同步控制端 LDN，故并没有发生提前预置的现象（即对于同步控制口，即使信号有毛刺，但只要毛刺不出现在时钟的有效边沿处，也不会出现误操作情况），但毛刺毕竟不是好现象，最好设法除去。因为若用这个信号去控制异步端口，则有可能发生误操作。

对于毛刺的解决方案有多种，比如可以使用高速 FPGA 来解决这些问题。因为此类毛刺出现的原因多数是由于器件速度低造成的。当器件的工作速率较低时，多个通道上同时出发的信号在经历了一段距离后相互间的相位差就会明显增加，当通道上数据的变化过大时尤其如此，如从 7 变到 8，即从 0111 变到 1000，每一个通道上的数据都发生了翻转。显然器件的高速性能越差，出现的毛刺的可能性就越大，毛刺的宽度也越大。

图 8-5 是基于 ACEX1K 系列 EP1K30TC144-3 目标器件综合后的时序仿真波形。

ACEX1K系列器件是Altera较早期推出的FPGA，速度性能还稍差。

如果选择目标器件为较新的Cyclone系列EP1C3T144C8，对应的仿真波形图则如图8-6所示，已经去除了部分毛刺。如果仍然基于相同的逻辑电路（图8-4），图8-7则是基于更高速的FPGA：Altera较新近推出的Cyclone Ⅲ系列的EP3C10E144C8，综合后的仿真波形显示，已经完全去除了所有的毛刺。

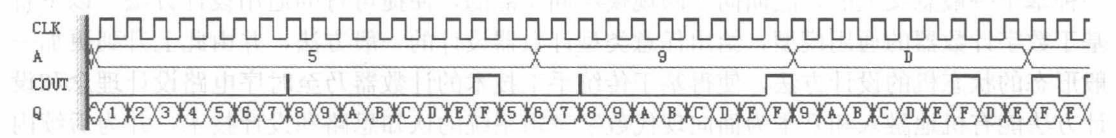


图8-6 图8-4的时序仿真波形（基于Cyclone系列EP1C3T144C8）

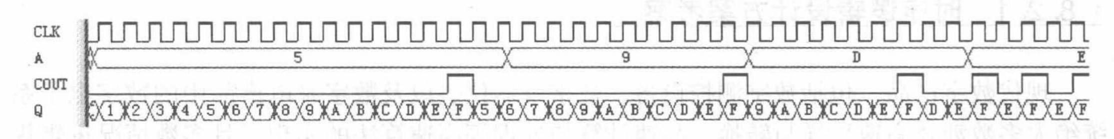


图8-7 图8-4的时序仿真波形（基于Cyclone Ⅲ系列EP3C10E144C8）

由以上的讨论可以推想，如果使用传统的74LS系列低速器件来构建电路，其毛刺情况一定更加难免，且难以发现，这将严重影响电路的工作可靠性。

并非所有存在毛刺的设计都与器件速度性能有关（此类电路在后文给出的示例中将会遇到）。对于由逻辑原因导致的毛刺，只有通过改变整体设计方案，选择不同的布线布局方案，选择不同的约束条件，或直接使用特定的附加电路才能有效地去除。图8-8给出了一种除毛刺的解决方案，即在预置控制信号通道上插一D触发器和反相器，使进位信号导致的预置信号延时半个时钟，从而避开了所有的毛刺。

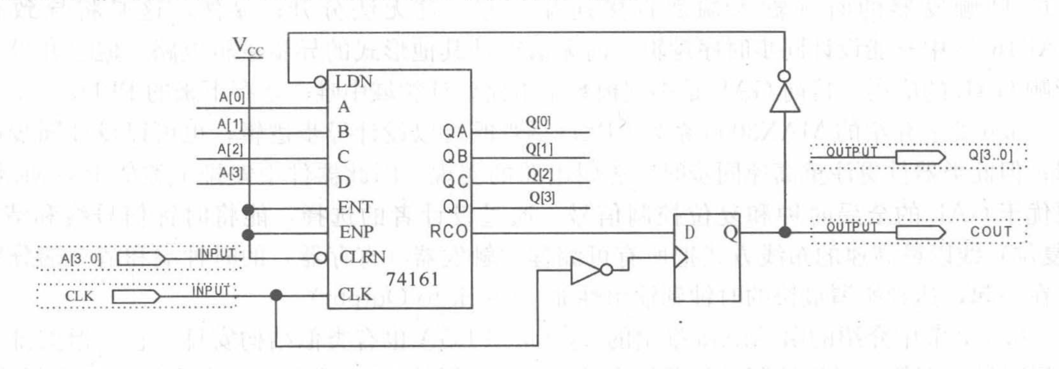


图8-8 利用74161的数据预置口构成的模可控型计数器

事实上，此节中介绍的计数器设计电路多数情况没有多少实用价值，通常现代时序逻辑电路的设计主要是利用HDL直接表述和设计的。本节的目的只是通过不同类型的电路设计，从某个侧面给出一些有普适意义的设计技术和设计经验。更多的内容和工程设计经验则需要读者通过大量的实验和设计实践去认识和积累。

8.2 计数器通用设计模型

第7章中基于传统数字技术介绍了常用类型的时序电路的分析和设计方法，特别是7.6节中还给出了使用专用集成器件完成的一些典型设计。但所有这一切最终并没有提供一种基于一般意义上的，能面向不同规模不同功能的，便捷可行的通用设计方法。以下将基于数字计数器的通用模型，给出任意类型计数器设计的一般方法，并由此上升到更加一般形态的状态机的设计方法，使得基于传统手工技术的计数器乃至时序电路设计理念和设计方法能有机地融入和升华为面向现代数字逻辑系统的认知思路和设计技术，并为后续内容的学习和实践做好准备工作。

8.2.1 时序逻辑设计方案考察

现代数字产品，包括数字测控设备、数字仪器仪表以及数字家电产品中的数字电子系统绝大多数涉及高速采样与转换、高速计算与处理等高速算法的实现，且多数情况逻辑规模巨大。此外，往往产品的高速性能与其综合性能几乎是并行的，这导致高速大规模数字系统的设计变成了十分普通和基本的设计目标。

显然，高速大规模数字系统设计技术成了数字电子技术中的重要组成部分。这使得传统的低速设计方法和方案被放弃，而纷纷采用一切可能的高速设计措施。从设计方案来说，最明显的莫过于一切时序逻辑设计都采用同步时序方案。于此相对应的是使硬件平台尽可能适应同步时序电路的设计。

从早期出现的GAL就可以看出其更适合于同步逻辑的设计。例如5.6.1节的图5-26中GAL16V8结构显示，它的时钟输入端只能是第1脚，此时钟将所有8个OLMC中的D触发器的时钟输入端都直接连在一起，且无法分开；显然，这必将导致在GAL16V8中只能设计同步时序逻辑，而无法设计其他形式的异步逻辑电路。但这并没有影响GAL的应用，目前GAL是小规模数字电路设计领域中唯一幸存下来的PLD。

5.6.2节介绍的MAX3000系列CPLD尽管既可以设计异步逻辑，也可以设计同步逻辑；但此类器件更注重高速同步时序逻辑电路的实现。因此器件中安排了类似于GAL但更优于GAL的全局时钟和复位控制信号。通过设计者的选择，能将时钟信号线和清0（复位）线以最高速的布线方式把所有可编程的触发器（寄存器）的时钟端和清0端分别连在一起，达到所谓最快的时钟到输出性能（Clock to Output）。

5.6.3节中介绍的由Altera推出的Cyclone FPGA也有类似结构安排。它不但安排了全局时钟（即能将时钟控制线与器件中的所有触发器的CLK端相连）和全局复位控制端，而且为提高同步时序逻辑的全局控制速度，安排了全局时钟网络，并为提高时钟信号的频率、信号波形质量和优化控制方式提供了一到多个嵌入式锁相环。

通过全局时钟构成的同步时序逻辑结构不仅能保证时序电路的高速性能，同时也能提高纯组合电路的速度。例如，可以在组合电路中加入同步时序的寄存器电路结构，构成所谓流水线结构，从而可以通过控制其中的寄存器的时钟频率来提高组合电路总体工作速

度。另外，还可以将比较器、乘法器等本来是纯组合电路的器件设计成多级流水线结构，通过控制其时钟频率，使得电路的工作速度比原来有成倍的提高。

基于以上讨论，以下给出的计数器的一般模型将定位于同步时序逻辑结构。

8.2.2 计数器的一般结构模型

结合图 7-1 及将要介绍的状态机结构，可以将同步计数器、序列检测器、序列发生器等模块，乃至一般时序电路，归结为如图 8-9（a）所示的一般结构模型。

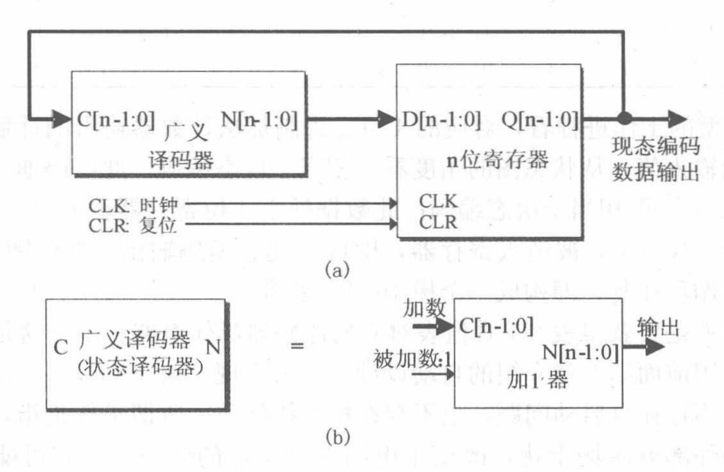


图 8-9 计数器的一般结构模型

这个结构模型主要由两个部件构成，一个是 n 位寄存器，这是一个纯时序电路，其内部结构类似于图 7-15（此图的 n=4）。CLK 是针对所有触发器的同步时序控制时钟，CLR 是针对所有触发器清 0 端的异步复位信号；另一个是广义译码器，是一个纯组合电路模块，与之对应的真值表的输入信号有 n 个，对应的输入信号 $C[n-1..0]=C_{n-1}$，…，C_1，C_0；输出信号也是 n 个，即 $N[n-1..0]=N_{n-1}$，…，N_1，N_0。

以下将据此一般设计模型，作为其特殊形态的应用，介绍几种经典计数器的设计方法。其中的任何一例都能很容易地推广到更大规模的设计。

以下各节将基于这里的计数器一般模型针对不同类型计数器的设计进行讨论。

8.2.3 普通二进制计数器设计讨论

若设计的是 4 位二进制计数器，则是模 16 的计数器。对于图 8-9，n=4。由于是从 0000 计到 1111。因此广义译码器的真值表应该如表 8-1 所示。此表的输入数据 C[3..0] 从 0000 到 1111；对应的输出数据 N[3..0] 比对应的 C[3..0] 多 1，构成计数累加方式。

从纯译码器的角度看，真值表 8-1 即为一个译码器真值表，其输出码是输入码的响应。例如输入码为 1010，对应的译码输出则是 1011。但这个译码器也等效于一个加常数 1 的 4 位加法器，见图 8-9（b），即输出数据 N 总是比输入数据 C 大 1。

表 8-1 加 1 器真值表

输入 C[3..0]				输出 N[3..0]				输入 C[3..0]				输出 N[3..0]			
C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0	C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	1	0	0	1	1	0	1	0
0	0	1	0	0	0	1	1	1	0	1	0	1	0	1	1
0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1	1	1	0	0	1	1	0	1
0	1	0	1	0	1	1	0	1	1	0	1	1	1	1	0
0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0

从图 8-9 模型的工作进程看，对应的 4 位二进制加法计数器输出的计数值是 C[3..0]，恰好是译码器的输入值，从状态图的角度看，相当于现态编码。此编码通过译码器，被加上 1，输出 N[3..0]，即相当于次态编码；此数据处于 4 位寄存器输入口。此时当 CLK 出现一个上升沿后，N[3..0] 被锁入寄存器，即成为现态编码输出。如此随着 CLK 时钟脉冲的不断出现而循环往复，即构成一个模 16 的计数器。

最后的任务就是要获得表 8-1 真值表对应的译码器具体电路。由于这是一个纯组合电路，完全可以使用前面第 6 章介绍的自动设计方法来实现。由于 C[3..0] 涉及 4 位二进制所有编码，所以不存在自启动问题，也不存在抗干扰问题。所谓干扰是指，如果在正常计数过程中，受到外部电磁场干扰，改变了电路中触发器的状态，则有可能飞到非法状态中，导致无法继续正常计数。

8.2.4 BCD 码计数器设计讨论

这里将使用图 8-9 的一般模型来设计例 7-3 提出的计数器。通过以上的讨论，此项设计就变得非常简单了，整个设计的关键就是根据题意为译码器配上合适的真值表。显然表 8-2 完全符合题意要求：它从 C[3..0]＝0000 开始到 1001 结束，对应此输入值 1001(9) 的输出值是 0000，从而回到最初的计数值，并且它们都属于 8421 BCD 码。

表 8-2 模 10 计数器真值表

输入 C[3..0]				输出 N[3..0]				输入 C[3..0]				输出 N[3..0]			
C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0	C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0
0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	0
0	0	0	1	0	0	1	0	0	1	1	0	0	1	1	1
0	0	1	0	0	0	1	1	0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	1	0	1	1	0	0	1	0	0	0	0

如果要考虑抗干扰和自启动问题，考察表 8-2 的真值表发现对应的电路不是唯一的，所以就不能保证一定能解决自启动问题。最好的方法就是将表 8-2 的真值表改为表 8-3，

这就能保证电路的自启动安全了。这是因为C[3..0]的1001以下的非8421码都被强行导入初始状态码0000，其对应的状态图如图8-10所示，从而确保了系统工作的可靠性。

表8-3 改进后的真值表

输入C[3..0]				输出N[3..0]				输入C[3..0]				输出N[3..0]			
C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0	C_3	C_2	C_1	C_0	N_3	N_2	N_1	N_0
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0	1	0	1	1	0	0	0	0
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	1	1	0	1	0	0	0	0
0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0

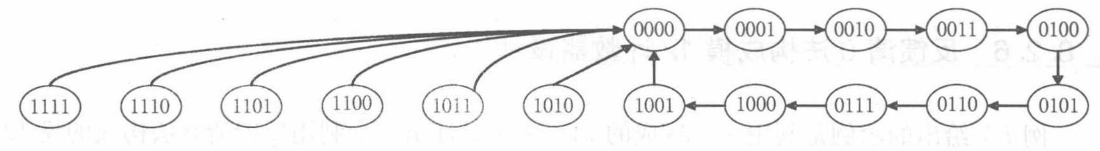


图8-10 表8-3真值表对应的状态图

8.2.5 模可控计数器设计讨论

利用图8-9的一般模型设计例7-6同样十分直观和简单，即首先根据题意和图8-9的模型，得此例电路模型结构图，如图8-11所示。此电路的3位寄存器的结构类似于图7-15。图8-11中的两个译码器分别对应两个计数模，上方和下方的译码器分别对应的真值表如表8-4和表8-5所示。这两个译码器的输出通过一个多路选择器。多路选择器当然也对应一张真值表，也是纯组合电路，这在第6章中已有详述，这里不再重复。

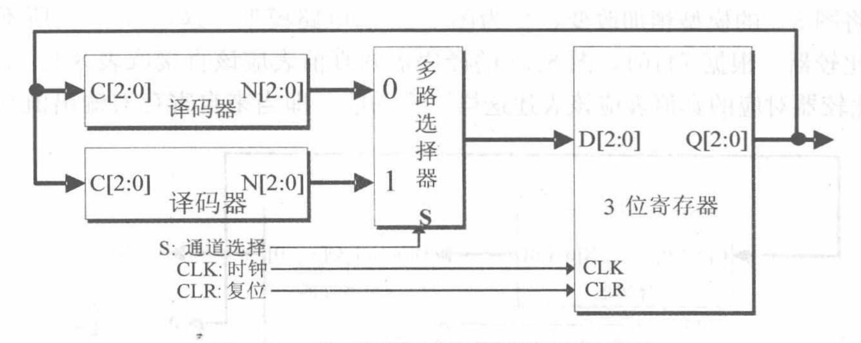


图8-11 模可控同步加法计数器电路模型

从图8-11可以看出，两张真值表输出的次态码需要通过一个多路选择器传给寄存器。多路选择器的控制端是S，当S=0，表明上方的译码器的数据可以通过选择器，对应模5

的计数器；否则，即 S=1，则表明下方的译码器的数据可以通过选择器，对应模 7 的计数器。分析表 8-4 和表 8-5 真值表可知，无论作为模 5 还是模 7 计数器，都不存在无法自启动的问题。

表 8-4 模 5 计数器的译码器真值表

输入 C[2..0]			输出 N[2..0]		
C_2	C_1	C_0	N_2	N_1	N_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

表 8-5 模 7 计数器的译码器真值表

输入 C[2..0]			输出 N[2..0]		
C_2	C_1	C_0	N_3	N_2	N_1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	0

8.2.6 反馈清 0 法构成模 12 计数器设计讨论

例 7-7 给出的示例是利用一个现成的 74LS161 设计出一个利用反馈清 0 法构成的模 12 加法计数器，但这里将利用一般模型自主设计一个相同功能的计数器。

事实上，如果仅仅是设计一个模 12 的加法计数器，只需对图 8-9 的模型选择表 8-1 的真值表，并对此表稍加改变，即直到 C[3..0] =1011 时，才让次态码返回初始状态（即 N[3..0]=0000)，对表中以下的编码不变。

显然，基于这种方法设计的模 12 计数器的可靠性要远远好于 7.6.1 节的电路（7.6.1 节的电路靠反馈控制构建)。因为整个计数流程是由图 8-9 中加 1 器所代表的状态译码器来确定的，与寄存器工作情况毫无关系，整个计数过程中即使有毛刺也肯定不会影响计数的可靠性。当然作为一个示例，这里仍然按照 7.6.1 节的方案，用图 8-9 的模型设计一个同类计数器。

首先将图 8-9 的模型稍加改变，变为图 8-12 的电路模型，这里 n=4。所不同之处是加了一个比较器。根据 74161，图 8-12 的译码器的真值表应该直接取表 8-1。由于是异步清 0，此比较器对应的真值表应该表述这样一个功能，即当来自寄存器输出值与“计数控

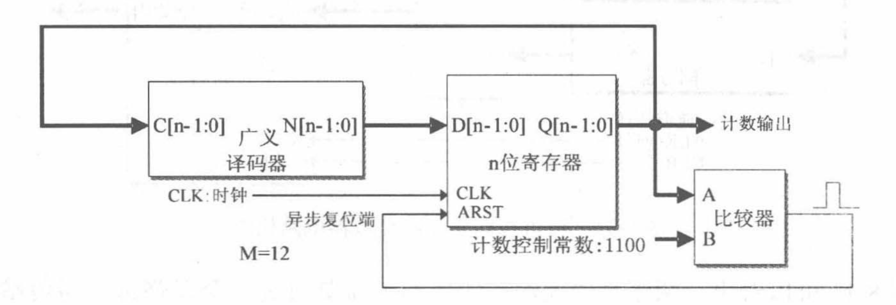


图 8-12 基于一般模型的反馈清 0 型模 12 加法计数器

制常数”端的数据相等时，比较器输出一个高电平（设此寄存器的异步清0端ARST高电平有效），对寄存器异步清0。如果要构成十二进制计数器，这应该在“计数控制常数”端放置常数1100。其实图7-36电路输出端接的与非门功能上就是这样一个比较器。

注意在实际情况中，类似于图8-12模型的设计（由于有反馈控制端），须通过时序仿真密切关注可能的毛刺脉冲对计数器工作的可靠性和可行性的影响。

8.2.7 同步加载型计数器设计讨论

图8-13是根据图8-9修改而来的，图中增加了一个比较器和一个多路选择器。比较器的功能和结构与图8-12相同。多路选择器是构成同步加载功能的重要器件，译码器对应的真值表仍然选择表8-1。

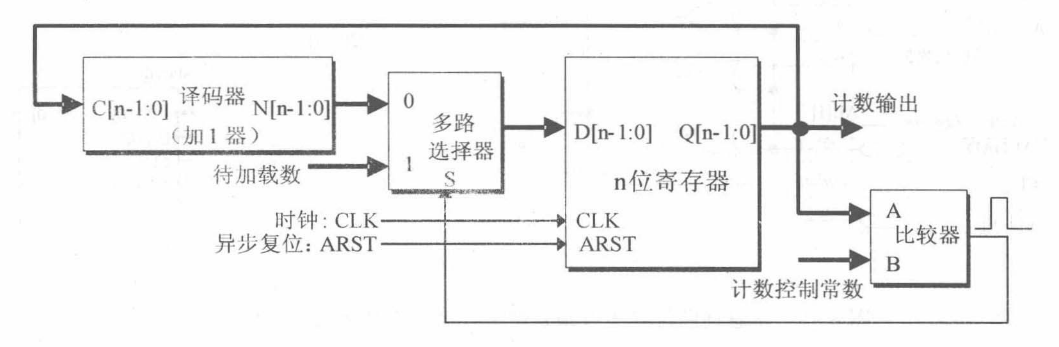


图8-13 基于一般模型的同步加载型计数器结构模型

图8-13的工作原理是这样的，当计数器的计数值计到等于“计数控制常数”时，如1011，比较器就输出1，使多路选择器选择S=1的通道，如果此时“待加载数”等于0000，则在紧接着的一个时钟脉冲后，此0000被锁入4位寄存器中，从而开始一个以0000为起始的新的计数周期。如果计数值小于“计数控制常数”时（此时S=0），则进行正常计数，这便是一个模12的计数器。如果改变“计数控制常数”和“待加载数”则能实现不同状态码区间的计数。所谓状态码区间是指计数状态的选择范围。例如模3计数器可能的状态区间有很多，对于4位计数器的编码可以有0000、0001、0010，或1101、1110、1111，或0100、0101、0110，等等。

8.2.8 异步加载型计数器设计讨论

基于一般模型的异步加载法构成的计数器应该如图8-14所示。图8-14的电路与图8-12相似，只是n（n=4）位寄存器稍有不同。这是因为异步置数必须使用触发器的异步清0和异步置位端来实现，这就需要一个特殊结构的寄存器来担任。图8-15（a）就是一个2位异步预置型寄存器ALCH2（图8-15（b））的内部电路原理性结构图。注意图8-14电路的功能与图8-13基本相同，只是前者是异步加载、后者是同步加载，计数中要相差1。

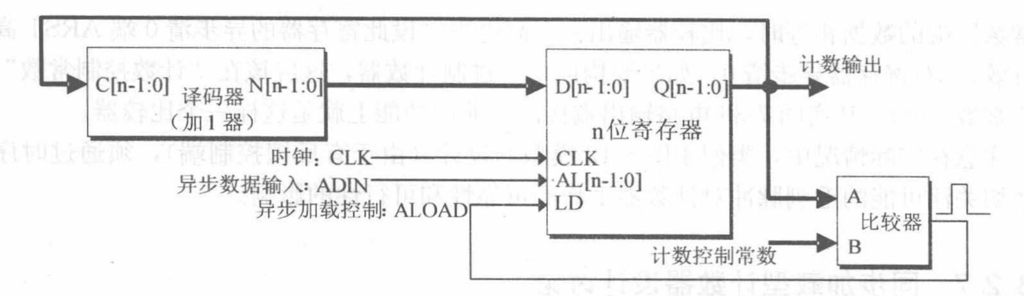


图 8-14　基于一般模型的异步加载型模 12 加法计数器结构模型

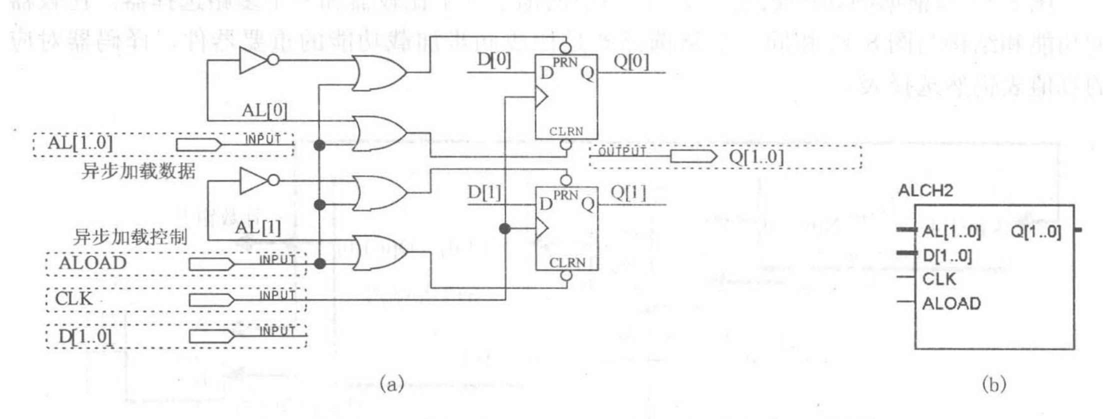


图 8-15　示意性两位异步可预置型寄存器电路结构及其元件符号

8.2.9　可逆计数器设计讨论

其实可以根据图 8-11 的模型很容易地把可逆计数器设计出来。图中的两个状态译码器，一个译码器的真值表构成加法计数器；另一个译码器则用减法真值表（这个真值表的设计留给读者）。多路选择器的选择端 S 就可以控制加/减。

从以上 6 个示例可以看出，各种不同类型不同功能的计数器，也包括其他尚未提到的同步时序逻辑类型，如移位寄存器、序列检测器等，其实都不过是基于图 8-9 所示的基本模型的简单变化；它们都脱离不了这样一个基本事实，即尽管是时序电路，但设计的关键和主要的工作却是组合电路。如图 8-13 所示的译码器、比较器、多路选择器都是组合电路，都可以用真值表来描述，而本质上都是广义译码器的不同的特殊表现形式。因此，就实用领域的数字设计而言，可以这样说，任何组合或时序逻辑电路的设计，其核心的内容都可归结为组合逻辑电路的设计，最终又可归结为一个含有特定意义的“译码器”的设计。这显然是一条数字逻辑电路设计的捷径！

从另一个角度看，本节以前给出的传统数字技术展示的一系列电路示例、设计方法或功能结构都不过是图 8-9 模型的特殊形式，这就是本节将其称之为一般模型的目的。而图 8-9 给出的模型本身，无非是有限状态机的一种特殊形式。

8.3 从计数器的一般模型到状态机

其实，从第 4 章开始，前面各章节介绍的内容完全可以看成是为本节服务的，是通往本节将要介绍的状态机概念的一个又一个逐级提升的台阶，是一个从特殊到一般的循序渐进的数字技术学习过程的展开和推进，也完全符合人类认识世界的基本规律。

以表 8-3 为例，从表面上看，这只是一张真值表，对应的电路模块只是一个等效于译码器的纯组合电路。然而如果把它与图 8-9 结合，完全可以把它看成是一个 BCD 码计数器的状态转换图；而这个计数器本身就是一个状态机，因为表中的输入数据（C_3，C_2，C_1，C_0）对应的是现态状态编码（如 0110），输出数据（N_3，N_2，N_1，N_0）对应的是次态状态编码（如 0111），它们与图 8-10 有直接的对应关系。当然，对于状态机而言，现态和次态是针对图 8-9 中的寄存器模块而言的。

显然，可以把图 8-9 中的译码器稍加改变就能得到如图 8-16 和图 8-17 所示的更具有一般意义的电路模型，即有限状态机的结构模型。而之前所讨论的所有电路结构应该是此模型的特殊形式，即不同类型的计数器只是状态机的特殊形式。

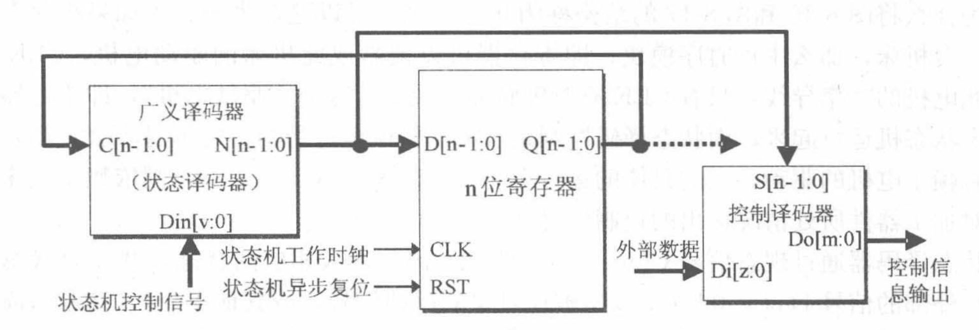


图 8-16 Mealy 型有限状态机一般模型

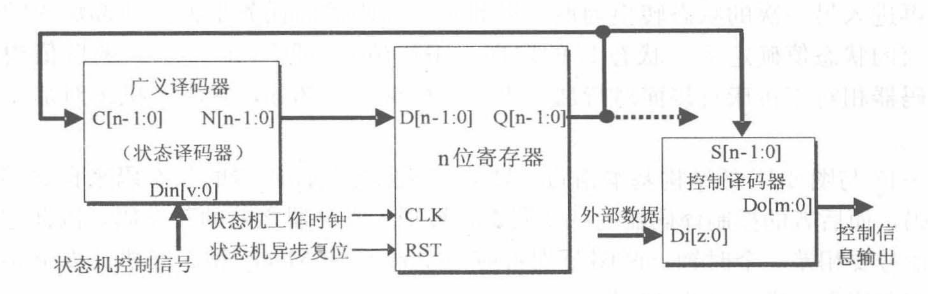


图 8-17 Moore 型有限状态机一般模型

如图 8-16 所示的典型的有限状态机由 3 个核心模块组成：一个是寄存器，是一个由触发器构成的简单的同步时序电路，也称为主控时序模块；另一个是纯组合电路构成的广义译码器，也是一个主控模块，在状态机结构中被称为状态译码器。此状态译码器的 C 和 N 分别是状态机的现态编码和次态编码的输入/输出口，译码器及其 C 与 N 同寄存器构成

了状态机的一个闭环运行体系，这是第一个闭环结构。在此闭环结构的工作中，次态编码N取决于现态编码值C，以及外部控制信号Din[v..0]。

第3个核心模块是控制译码器，负责对外部控制对象输出控制信号，控制一切适合于状态机控制的对象，如步进电机、温度测控设备、通信接口、自动化机械、智能家电等。控制译码器根据来自状态机的状态码，从控制对象读入需要的数据Di[z..0]或向控制对象发出控制信息Do[m..0]，而被控对象则将自身的状态信息（如温度是否超限）通过Din[v..0]进入状态译码器，以便使状态译码器及时调整状态走向，从而通过控制译码器改变控制信息Do[m..0]，始终使控制对象处于正常工作状态。这时状态译码器及其Din[v..0]、控制译码器及其Do[m..0]，以及被控制对象构成了第二个闭环结构。

当然也有开环形式的控制情况。

控制译码器也是组合电路。如果其输出的对外控制信号和数据有时需要进行锁存或存储，则应该在输出的Do[m..0]端口上加上锁存器或存储器等。

图8-16和图8-17的寄存器输出口用了虚线箭头，表示有些情况下可以将状态码C[n−1..0]直接作为对外的控制信号或数据输出，如各类计数器、A/D的采样控制等（这时的状态码不一定是常见的累加式编码），这时就可省去控制译码器了。

为什么将图8-16和图8-17的结构称为状态机呢？可以这样来理解，如果将状态机比喻为一台机床，那么主控时序模块，即寄存器可以被看成此机床的驱动电机，CLK信号即为此电机的功率导线，只有CLK有时钟脉冲出现时，这个“驱动电机”，即寄存器才能使这个状态机运行起来；而状态译码器则是机床的机械运行结构，它本身不会转动，它的运转有赖于电机的驱动，它的具体的运行走向（次态编码N[n−1..0]）则依赖于机床操作者针对加工器件所处情况得出的控制信息（Din[v..0]）。

状态译码器通过现态信号C[n−1..0]的状态码，进入相应的状态，并在此状态中根据来自外部的信号Din[v..0]（可以是被控对象的反馈信号，或其他指令），结合当前状态码C[n−1..0]确定下一状态的走向，即机床的下一步动作，即向次态信号N[n−1..0]中赋入相应的状态值（状态码）。此状态码将通过N口传给寄存器，直至下一个时钟脉冲的到来再进入另一次的状态转换周期。因此状态译码器的任务是根据外部输入的控制信号和当前的状态值确定下一状态N的取向（编码值），即N[n−1..0]的取值内容；而控制译码器相对于机床直接面对待加工工件的机械加工部分，它决定机床对加工工件的工作方式。

图8-17与图8-16的结构基本相同，只是前者的控制译码器的状态码来自寄存器输出的状态码，而后者的控制译码器的状态码来自被锁入寄存器之前的状态码，因此它们输出的控制信号要相差一个时钟；两图分别对应Moore型和Mealy型状态机。以下将用具体示例详细介绍状态机的设计和应用。

8.4 基于一般模型结构的计数器设计

这里将给出基于8.2节的计数器一般模型的计数器设计具体示例。由8.3节可知，同步计数器本质上只是状态机的一种特殊形式。因此对于此类计数器的理解和设计，最好的

把握角度是对状态机的清晰理解。

8.4.1 基于一般模型的十进制计数器设计

图 8-18 是根据图 8-9 的模型编辑的十进制计数器电路原理图。其中的元件 DFF4 是 4 个 D 触发器构成的 4 位锁存器，内部结构如图 6-44 所示；CNT10 模块是基于广义译码器真值表的 case 语句程序构成的元件，其 Verilog 代码表述如图 8-19 所示。

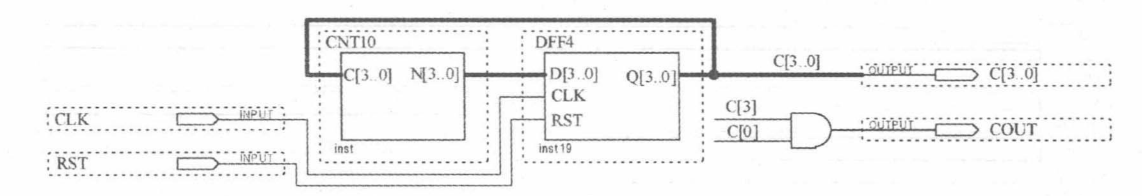


图 8-18 十进制计数器电路原理图

由此程序及图 8-18 电路可知，当 DFF4 的输出 C[3..0] 为 1001，即 9 时，进位信号 COUT＝1，且在 CLK 的下一时钟信号后，计数值输出将回到初值：0000。显然，这是一个十进制计数器。在 Quartus Ⅱ 上编译后，如果没有错，可以得到如图 8-20 所示的硬件资源利用报告。报告表明，此项设计中，总逻辑宏单元的占用数是 5 个，其中组合模块 5 个，触发器 4 个。图 8-21 是此计数器的时序仿真波形。波形表明，其计数值范围是 0～9，在 7 与 8 之间有毛刺，这个毛刺对计数器本身没有任何影响；RST 负脉冲复位。

图 8-22 是图 8-21 波形的展开情况。通过图 8-22 可以了解到信号的延时情况。图中，时钟信号 CLK 的上升沿处于 9.5us（这里的 u 即 μ）处，而响应此时钟的计数值 8，直到 9.506us 处才出现。即这时，原来的计数输出 7 才转变为 8。而在 7 与 8 转变处，4 位

```
1  module CNT10 ( C,N );
2      input [3:0] C ;
3      output [3:0] N ;
4      reg [3:0] N;
5    always @ (C,N)
6      case( C )
7      4'b0000 : N<=4'b0001;
8      4'b0001 : N<=4'b0010;
9      4'b0010 : N<=4'b0011;
10     4'b0011 : N<=4'b0100;
11     4'b0100 : N<=4'b0101;
12     4'b0101 : N<=4'b0110;
13     4'b0110 : N<=4'b0111;
14     4'b0111 : N<=4'b1000;
15     4'b1000 : N<=4'b1001;
16     4'b1001 : N<=4'b0000;
17     default : N<=4'b0000;
18     endcase
19   endmodule
```

图 8-19 元件 CNT10 的程序

Family	Cyclone III
Device	EP3C10E144C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	5 / 10,320 (< 1 %)
Total combinational functions	5 / 10,320 (< 1 %)
Dedicated logic registers	4 / 10,320 (< 1 %)
Total registers	4
Total pins	7 / 95 (7 %)
Total virtual pins	0
Total memory bits	0 / 423,936 (0 %)

图 8-20 计数器资源使用报告

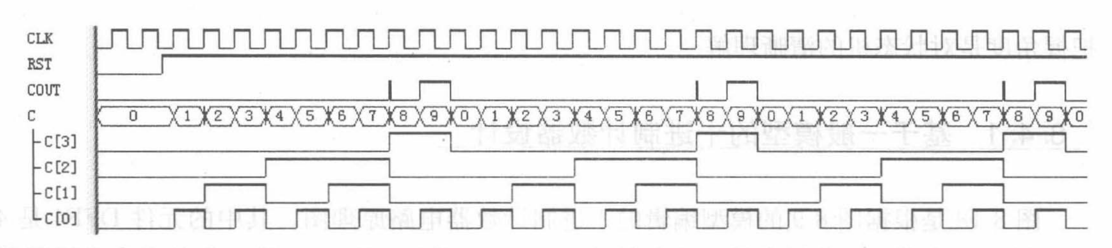


图 8-21 图 8-18 电路的时序仿真波形

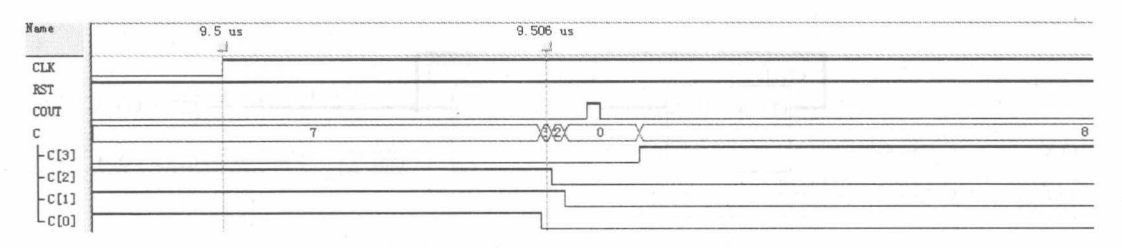


图 8-22 观察电路信号的延时情况

二进制数据 C[3..0] 的变化时间并不一致，这显然是由于每一数据位通过的门电路的延时时间不同造成的。它们从 7 变到 8 之间经过了其他数据的快速变化（如 2、0 等），因此在这里容易产生毛刺脉冲（当然其他地方也可能会不同程度地出现毛刺现象）。

8.4.2 含自启动电路的十进制计数器的设计

在第 7 章中已多次提到，实用计数器的设计中，除了保证电路能正确计数外，还必须检验并保证它的自启动性能。显然，图 8-19 的程序是无法确保自启动的，因为它没有考虑一旦计数器的计数值进入到 10～15 时电路该如何操作。为此必须进行改进。图 8-23 是对图 8-19 程序中 case 语句部分的数据的改进方案之一，方法简单直观。这种改进方式确能解决计数器的自启动问题。此程序的特点是，一旦进入 6 个非法计数值 1010～1111 中任何一个值时，经过数个时钟后，最终都将进入计数初值 0000。

```
case( C )
  4'b0000 : N<=4'b0001 ;
  4'b0001 : N<=4'b0010 ;
  4'b0010 : N<=4'b0011 ;
  4'b0011 : N<=4'b0100 ;
  4'b0100 : N<=4'b0101 ;
  4'b0101 : N<=4'b0110 ;
  4'b0110 : N<=4'b0111 ;
  4'b0111 : N<=4'b1000 ;
  4'b1000 : N<=4'b1001 ;
  4'b1001 : N<=4'b0000 ;

  4'b1010 : N<=4'b0000 ;
  4'b1011 : N<=4'b1010 ;
  4'b1100 : N<=4'b1011 ;
  4'b1101 : N<=4'b1100 ;
  4'b1110 : N<=4'b1101 ;
  4'b1111 : N<=4'b1110 ;
```

图 8-23 可自启动的“真值表”

相比之下，具有自启动功能的计数器的可靠性高，防干扰能力强。因为如果遇到强电磁干扰时，计数器即使在正常计数情况下，也有可能被干扰而导致触发器翻转，从而进入非法计数值。这时如果有图 8-23 代码对应的电路安排，就能很快进入正常计数循环。这种性能对于由此类电路结构构成的状态机尤其重要，因为状态机的抗干扰能力是一个重要的技术指标。

8.4.3 异步控制型任意模计数器设计

与 8.2 节给出的计数器设计方案不同，图 8-24 所示的模 N（在此 N 小于等于 16）计数器采用了 8.2.6 节的模型。图 8-24 中，译码器模块 CNT4BIT 与右面的 4 位锁存器构成了一个 4 位二进制计数器（CNT4BIT 的 case 语句表述如图 8-25 所示，程序其余部分与图 8-19 相同）。图最右侧的比较器模块 COMP2 决定模 N 的具体数值。如果是 12，其 case 语句表示如图 8-26 所示。

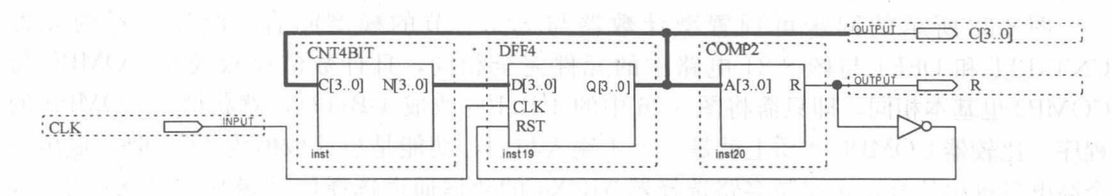


图 8-24 另一种形式的计数器电路结构

```
case( C )
 4'b0000 : N<=4'b0001 ;
 4'b0001 : N<=4'b0010 ;
 4'b0010 : N<=4'b0011 ;
 4'b0011 : N<=4'b0100 ;
 4'b0100 : N<=4'b0101 ;
 4'b0101 : N<=4'b0110 ;
 4'b0110 : N<=4'b0111 ;
 4'b0111 : N<=4'b1000 ;
 4'b1000 : N<=4'b1001 ;
 4'b1001 : N<=4'b1010 ;
 4'b1010 : N<=4'b1011 ;
 4'b1011 : N<=4'b1100 ;
 4'b1100 : N<=4'b1101 ;
 4'b1101 : N<=4'b1110 ;
 4'b1110 : N<=4'b1111 ;
 4'b1111 : N<=4'b0000 ;
 default : N<=4'b0000 ;
```

图 8-25 CNT4BIT 的 case 语句部分

```
module COMP2 (A,R);
  input [3:0] A ;
  output R; reg R;
 always @ (A,R)
  case( A )
  4'b1100 : R<=1'b1;
  default : R<=1'b0;
   endcase
endmodule
```

图 8-26 COMP2 的程序

图 8-27 是此电路图的时序仿真波形图。由图可见，其计数范围是 0～B。由于 COMP2 的输出控制了计数器的异步清 0 端，因此在计数到达 1100 时的一瞬间，COMP2 的输出就

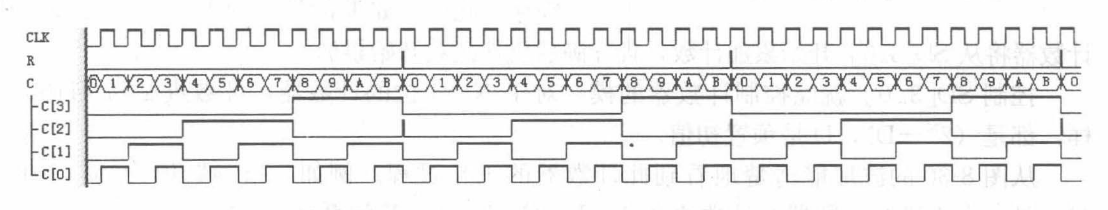


图 8-27 图 8-24 电路的时序仿真波形

将元件 DFF4 清 0 了，从而使计数器复位。这从波形图可以看出，信号 R 的宽度极窄，刚能维持将 DFF4 复位。如果 R 是接在同步端口上，就不是这种情况了。

8.4.4 初值可预置型计数器设计

与其他类型的计数器相比，计数初值可预置型计数器的适用面更宽。以上曾介绍的 74161 就是一个计数初值可同步预置型的计数器。本节介绍的计数器的功能与 74161 类似，只是使用了计数器一般模型的概念来设计。

图 8-28 所示的同步可预置型计数器与 8.2.7 节的模型吻合。此计数器的元件 CNT4BIT 和 DFF4 与图 8-24 电路中的元件完全相同，且计数值比较模块 COMPF 与 COMP2 也基本相同，即只需将图 8-26 中的 4′B1100 改成 4′B1111，就获得了 COMPF 的程序。比较器 COMPF 本质上就是一个 4 输入与门，功能是当计数值为 1111 时，输出一个高电平进位信号，它控制多路选择器 MUX4 的数据通道选择信号端 S。MUX4 的 case 语句程序表述如图 8-29 所示，也是一个广义译码器真值表的表述。

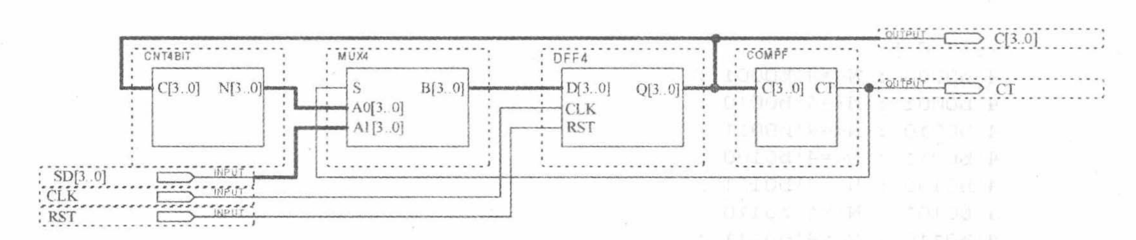


图 8-28 4 位同步可预置式 N 进制计数器

```
module MUX4 ( S, A0, A1, B );
   input S ; input [3:0] A0,A1;
   output [3:0] B ;
   reg [3:0] B;
  always @ (S,A0,A1,B)
   case( S )
    1'b1 : B<=A1 ;
    1'b0 : B<=A0 ;
  default : B<=4'b0000 ;
   endcase
 endmodule
```

图 8-29 元件 MUX4 的描述

由图 8-28 和 MUX4 的结构可以了解到，当计数器尚未计到 1111 时，S=0，此计数器处于正常计数操作，即 CNT4BIT 的输出通过 MUX4 进入 DFF4 的输入端；此后随着时钟的连续出现，进行正常的累加计数。

一旦计数器计到 1111 时，则 S=1，外部输入的预置值 SD[3..0] 将通过 MUX4 的 A1 口，进入 DFF4 的输入端 D[3..0]。若此时 CLK 出现了一个上升沿，则预置数据 SD[3..0] 被 DFF4 锁存。此后，如果预置数 SD[3..0] 不改变，则计数器将从 SD[3..0] 开始累加计数，且计满后仍从此数开始累加。

控制 SD[3..0] 就能控制计数器的模。对于 N=4 位的计数器，计数进制与 74161 一样，都是 (2^N-D)，D 是预置初值。

从图 8-30 的波形能清楚地看到此计数器的工作过程。例如，当预置值为 A (1010) 时，是一个六进制计数器，计数值在 A、B、C、D、E、F 间循环。

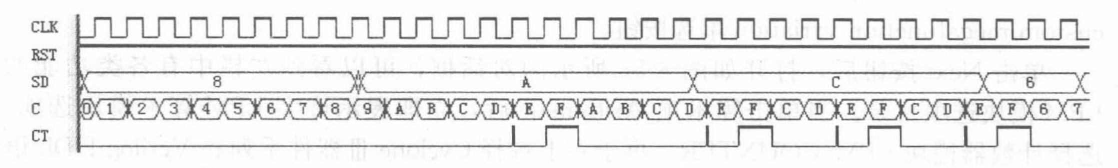


图 8-30 图 8-28 电路的时序仿真波形（目标器件是 Cyclone 系列 EP1C3T144C8）

从图 8-30 的波形还能看出，这是一个不可靠的计数器，因为预置数加载控制的进位信号 CT 有毛刺脉冲，且在时钟上升沿附近。好在这个计数器属于同步加载允许型的，狭窄的毛刺脉冲很难导致误操作。

其实，对于此类电路，图 8-30 中的毛刺是容易消除的，只要选择更高速的目标器件就可以了，如选择 Cyclone Ⅲ系列的 EP3C10E144C8 FPGA。

与 74161 一样，此类可预置型计数器同样可以用作分频比率可数控的分频器。

8.5 基于 LPM 宏模块的计数器设计

本节将介绍另一种利用自动设计技术更为便利地设计计数器的方法。即利用 Quartus Ⅱ 平台中现成的 LPM 宏功能模块来设计，并由此引出基于 LPM 模块的许多其他实用数字模块的自动设计技术。利用 LPM 这类宏模块实现高质高效数字电路功能模块的设计是数字系统自动设计技术的重要组成部分。

LPM 是 Library of Parameterized Modules（参数可设置模块库）的缩写，Altera 提供的可参数化宏功能模块和 LPM 函数均基于 Altera 器件的结构做了优化设计。在许多设计中，必须利用宏功能模块才可以使用一些 FPGA 器件中特定模块的硬件功能。例如各类片上存储器、DSP 模块、LVDS 驱动器、嵌入式锁相环 PLL 模块等。这些可以以原理图形模块或 HDL 硬件描述语言模块形式方便调用的宏功能块，使得基于电子设计自动化技术的效率和可靠性有了很大的提高。设计者可以根据实际电路的设计需要，选择 LPM 库中的适当模块，并为其设定适当的参数，就能满足自己的设计需要，从而在自己的项目中十分方便地调用优秀的电子工程技术人员的硬件设计成果。LPM 功能模块内容丰富，每一模块的功能、参数含义、使用方法等都可以在 Quartus Ⅱ 的 Help 中查阅到，即选择 Help 菜单中的 Megafunctions/LPM 选项。

本节具体介绍 LPM 计数器 LPN_COUNTER 的调用和测试的流程，给出 MegaWizard Plug-In Manager 管理器对宏模块的一般使用方法。对于之后介绍的其他模块的应用则主要介绍调用方法上的不同之处和不同特性的仿真测试方法。

LPM 模块的调用和参数设置步骤如下：

(1) 打开宏功能块调用管理器。首先根据 6.2 节，创建一个原理图工程（例如取此工程名为：CNT4BIT）。进入原理图编辑窗，于此编辑窗内任何一点双击，将弹出一个逻辑电路器件输入对话框，如图 6-7 所示。单击左下侧的按钮 MegaWizard Plug-In Manager，打开如图 8-31 所示的对话框，选中 Create a new custom megafunction variation 单选按钮，即定制一个新的模块。如果要修改一个已编辑好的 LPM 模块，则选中 Edit an existing

custom megafunction variation 单选按钮。

单击 Next 按钮后，打开如图 8-32 所示的对话框，可以看到左栏中有各类功能的 LPM 模块选项目录。当单击算术项 Arithmetic 后，立即展示许多 LPM 算术模块选项。选择计数器模块 LPM_COUNTER。再于右上选择 Cyclone Ⅲ 器件系列和 Verilog HDL 语言方式。最后键入此模块文件存放的路径和文件名，如：d:\LPM_MD\CNT4B，单击 Next 钮。

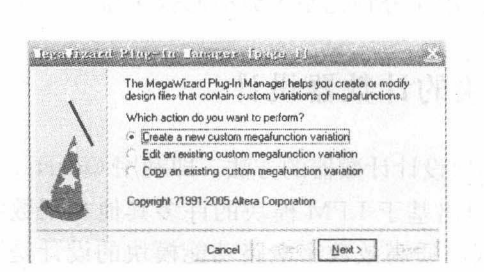


图 8-31　定制新的宏功能块

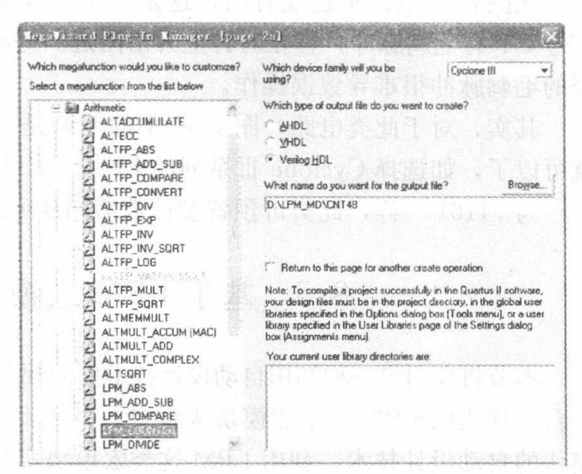


图 8-32　LPM 宏功能块设定

（2）单击 Next 按钮后打开如图 8-33 所示的对话框。在对话框中选择 4 位计数器，选择“Create an updown input…”使计数器有加/减控制功能。

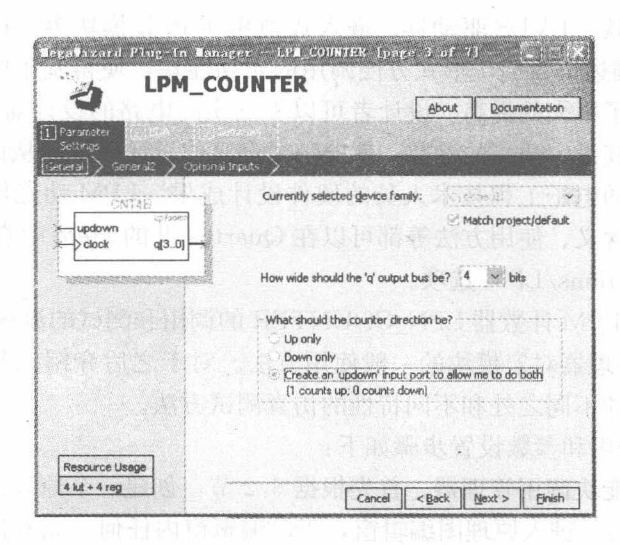


图 8-33　设 4 位可加减计数器

（3）再单击 Next 按钮，打开如图 8-34 所示的对话框。在此若选择 Plain binary 则表示是普通二进制计数器；现在选择 Modulus…12，即模 12 计数器，从 0 计到 11（BH）。

然后选择时钟使能控制 Clock Enable 和进位输出 Carry-out。

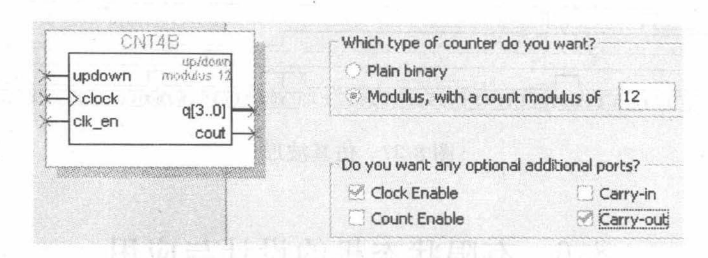


图 8-34 设定计数器，含时钟使能和进位输出

(4) 再单击 Next 按钮，打开如图 8-35 所示的对话框。在此选择 4 位数据加载控制 Load 和异步清 0 控制 Clear。最后再按 Next 按钮后就结束设置。以上流程设置生成了 LPM 计数器 CNT4B，可被调入当前原理图编辑窗中使用。

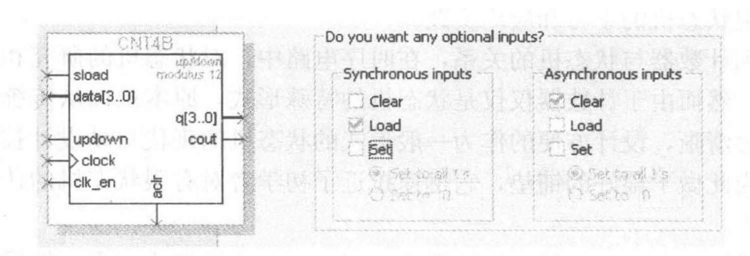


图 8-35 加入 4 位并行数据预置功能

完成后将此 LPM 模块调入原理图编辑窗，连接好引脚后，计数器电路如图 8-36 所示。图 8-37 是其仿真波形，从波形中可以了解此计数器模块的功能和性能。注意第二个 SLD 加载信号在没有 CLK 上升沿处发生时，无法进行加载，显然 SLD 是同步控制信号。根据此波形图，把详细讨论此计数器功能的任务留给读者。

其他的 LPM 元件的参数、功能设置和调用方法也类同，读者不妨多作一些练习。

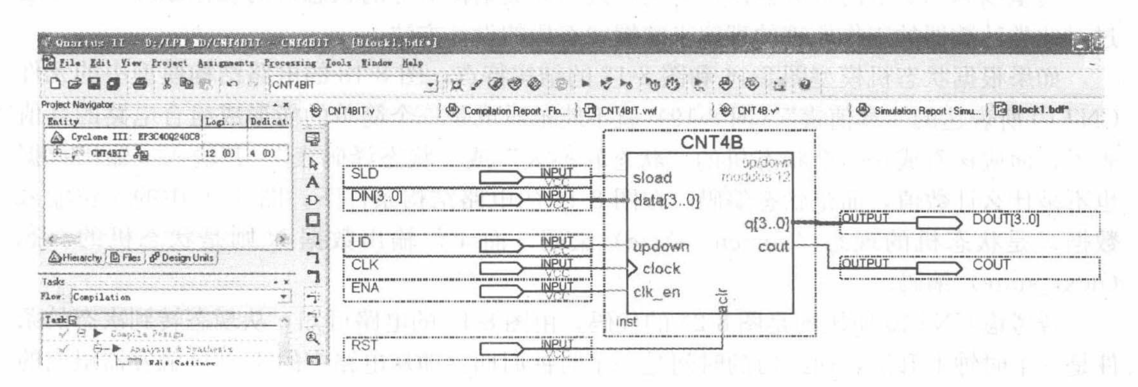


图 8-36 CNT4BIT 工程的 4 位计数器电路原理图

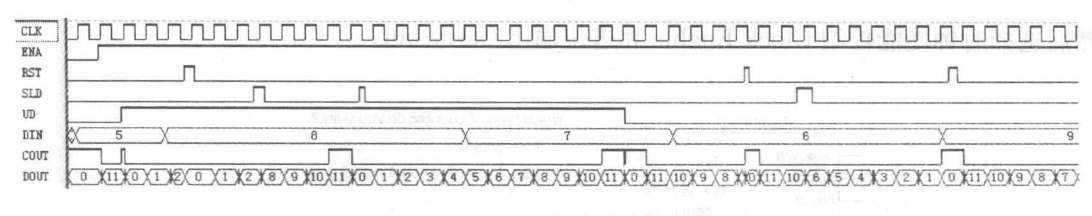


图 8-37 仿真波形

8.6 有限状态机的设计与应用

相比于 7.3 节介绍的用手工方法设计计数器的技术，8.4 节通过计数器的一般模型，利用专业自动化设计软件来设计和测试计数器，其先进的程度和极大的便利性是无可比拟的。然而，8.4 节借以各种示例向读者展示出的基于一般模型的计数器设计方法仍并非是本教材的最终目的，或最主要的目的。更具普适意义和更实用的数字电路设计技术的学习，就是对有限状态机的学习和应用实践。

前文曾提到计数器与状态机的关系，在时序电路中，对状态机的研究和应用更具有一般性和实用性，然而由于计数器仅仅是状态机的特殊形式，原本难以从传统的手工数字技术中归纳出概念清晰，设计方便的作为一般形式的状态机的现代自动设计技术。然而，以上各节的内容为此做了很好的铺垫，它迅速拉近了初学者对有限状态机的认识、学习和设计应用间的距离。

本节首先通过一个具体实例讨论计数器与有限状态机的对应关系，然后通过数个具有实用意义的时序电路设计示例展示面向不同应用角度的有限状态机的功能特点和设计方法(其实，8.4 节的内容也可以看成是状态机的应用示例)，并期待这些实例能够帮助打开读者的思路，使之能向更多其他的实用数字电路设计项目拓展状态机的应用。

8.6.1 计数器与状态机的对应关系

这里以 8.4.1 节的计数器设计为例，说明此类结构对应的状态机的工作原理，以及通过对此类计数器的工作原理的理解来掌握状态机的设计方法。

如果根据状态机模型图 8-16 和图 8-17 的结构特点，图 8-18 的电路结构模型中的元件 CNT10 所表达的“真值表”(图 8-19) 就不能被看成是一个简单的能形成组合电路的真值表了，而应该看成是一个状态机的“状态转换表”或“状态译码表”。其输入或输出数据也不是什么计数值，而是状态编码。在图 8-18 的电路结构条件下，图 8-19 中的 4 位输入数据 C 是状态机的现态 (current_state) 编码；而 4 位输出数据 N 则是状态机的次态 (next_state) 编码。

若考虑 CNT10 使用的是图 8-23 的代码，由图 8-19 的电路可知，从现态转到次态的条件是一个时钟上升沿，而经历的时间是一个时钟周期。即从电路中的 4 个 D 触发器组成的锁存器输出口 C[3..0] 从现态到次态的时间是一个时钟脉冲周期。

即如图 8-23 的代码所示，如果 C[3..0] 从出现 0001 到出现 0010 的间隔是一个时钟周

期。这个过程中，C[3..0] 首先出现的是 0001，这是当前现态编码，而 0010 则是次态编码。所以，之后的输入数据都是状态机的现态编码；“<=”之后的输出数据都是此状态机的次态编码。如果将图 8-19 看成是一个有 10 个状态的状态机，那么它有 10 个合法状态，6 个非法状态。这 16 个状态分别用 S_0、S_1、S_2、S_3、…、S_{15} 表示，它们分别对应的状态编码是 0000、0001、0010、…、1110、1111。根据图 8-23 的代码，可以获得此状态机的状态转换图，如图 8-38 所示。

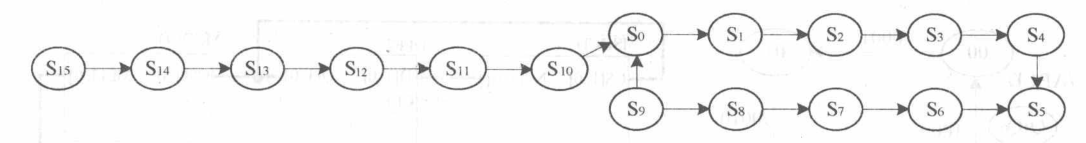


图 8-38　图 8-19 和图 8-23 对应的状态转换图

显然，此类基于一般模型的计数器与状态机有很直接的对应关系。

8.6.2 步进电机控制电路设计

1. 步进电机原理简介

步进电机作为一种电脉冲至角位移的转换元件，由于具有价格低廉、易于控制、无积累误差和计算机接口方面等优点，在机械、仪表、工业控制等领域中获得了广泛的应用。采用 PLD 控制步进电机十分常用和方便。利用 FPGA 能同步产生多路控制脉冲，可对多相多个步进电机进行灵活的控制。

步进电机的控制驱动是靠给步进电机的各相励磁绕组轮流通以电流，实现步进电机内部磁场合成方向的变化来使步进电机转动的。设步进电机有 A、B、C、D 四相励磁绕组，如图 8-39（a）所示，当同时有 4 个如图 8-39（b）所示的控制脉冲进入步进电机，使之产生旋转磁场，其中每两相产生一个合成磁场，为步进电机提供旋转动力。当给步进电机的 A、B、C、D 四相轮流通电时，其内部磁场的驱动方向就是 A→B→C→D，即磁场产生了旋转。当步进电机的内部磁场变化一周（360°）时，电机的转子转过一个齿距。

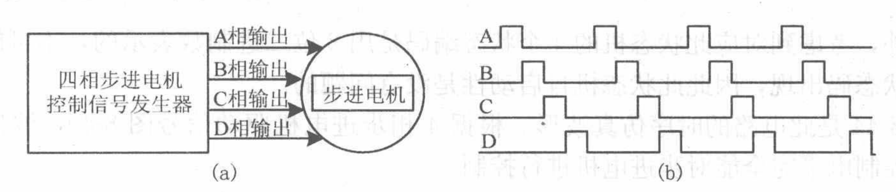


图 8-39　步进电机控制模型和控制时序

现在的问题就是设计一个时序电路能同步生成如图 8-39（b）所示的阶梯状脉冲信号。若此脉冲信号如图 8-39（b）中那样，以 A 相最先出现，其他各相依次出现，就能使电机向一个方向连续旋转，反之则向反向旋转。

2. 步进电机单向旋转控制电路设计

由于需要有 4 路不同相位的脉冲序列同步生成的控制电路，可以用一个 4 状态的状态

机来生成，其状态转换图如图 8-40 所示。根据 8.3 节的状态机模型，对应的单向旋转控制电路如图 8-41 所示，这是一个 4 状态状态机。其中的状态译码器模块 MSDCD 的程序如图 8-42 所示，程序中的现态码变量是 CS，次态码变量是 NS；控制译码器模块 MCDCD 的程序如图 8-43 所示，输出的 4 位 ML 信号用于控制电机旋转；图 8-41 中模块 DFF2 的内部结构与 DFF4 的图 6-44 电路类似，只是它是由两个 D 触发器构成的 2 位寄存器。

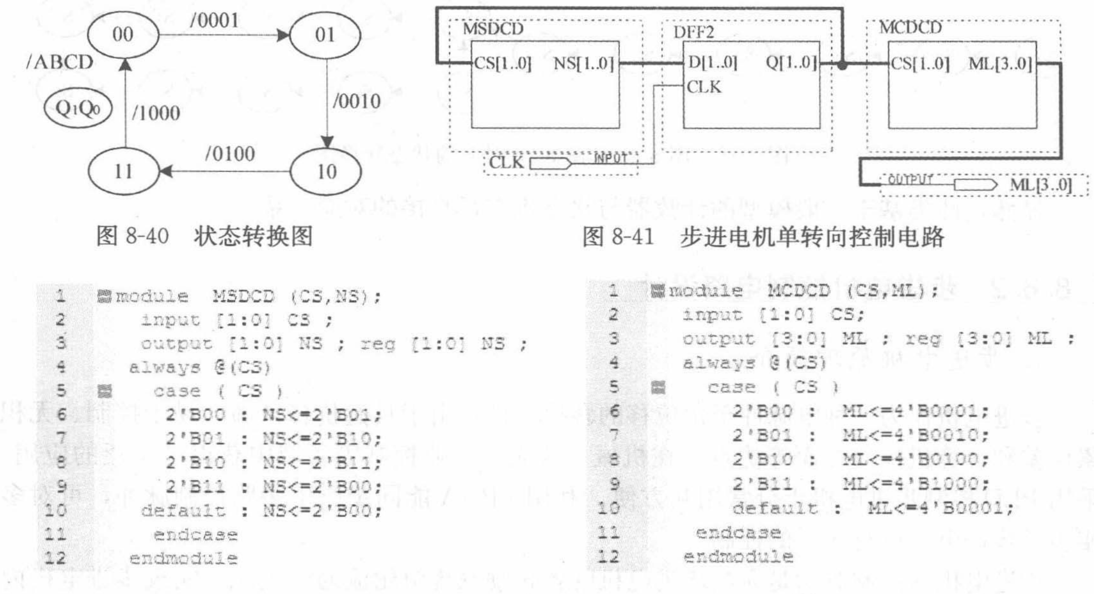


图 8-40　状态转换图　　　　图 8-41　步进电机单转向控制电路

```
1  module  MSDCD (CS,NS);
2    input [1:0] CS ;
3    output [1:0] NS ; reg [1:0] NS ;
4   always @(CS)
5     case ( CS )
6      2'B00 : NS<=2'B01;
7      2'B01 : NS<=2'B10;
8      2'B10 : NS<=2'B11;
9      2'B11 : NS<=2'B00;
10   default : NS<=2'B00;
11     endcase
12  endmodule
```

图 8-42　模块 MSDCD 的程序

```
1  module  MCDCD (CS,ML);
2    input [1:0] CS;
3    output [3:0] ML ; reg [3:0] ML ;
4    always @(CS)
5      case ( CS )
6        2'B00 :  ML<=4'B0001;
7        2'B01 :  ML<=4'B0010;
8        2'B10 :  ML<=4'B0100;
9        2'B11 :  ML<=4'B1000;
10       default :  ML<=4'B0001;
11    endcase
12   endmodule
```

图 8-43　模块 MCDCD 的程序

从图 8-40 和程序图 8-43 可以看出，在 4 个状态的任一状态中，恰好同步生成对步进电机输出的某一相的控制电平 ML[3..0]（类似于图 8-17 的 Do[m..0]），其高电平依次向高位步进。例如在状态 00 时，输出 ML[3..0]＝0001，状态 01 时，输出 ML[3..0]＝0010，如此等等。

另外，考虑到对应此状态机的 4 个状态编码是用 2 位二进制数表示的，不可能再有其他非法状态码出现，因此此状态机自启动性是没有问题的。

图 8-44 是此电路的时序仿真波形。根据 4 相步进电机驱动信号图 8-40，波形图 8-44 给出的控制电平完全能对步进电机进行控制。

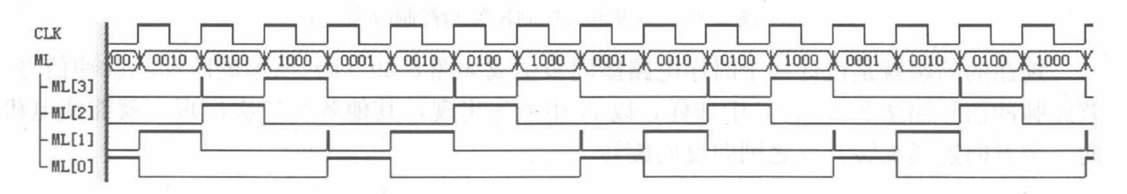


图 8-44　图 8-41 电路时序仿真波形

对于图 8-44 的输出信号中的毛刺脉冲，并不会影响电机的运行，这是因为步进电机

是电磁性负载，对含有狭窄脉冲的信号具有滤波作用。但如果输出的信号用作驱动其他电路，则不能保证一定安全。若为它用，可以增加一些电路，将毛刺除去。

图 8-45 是改进后的电路，其实就是在图 8-41 电路的基础上在模块 MCDCD 的输出信号 ML 端增加一个 4 位锁存器 DFF4，其锁存边沿恰好选择在此状态机工作时钟的下降沿，从而避开了毛刺。此改进电路的仿真波形如图 8-46 所示。

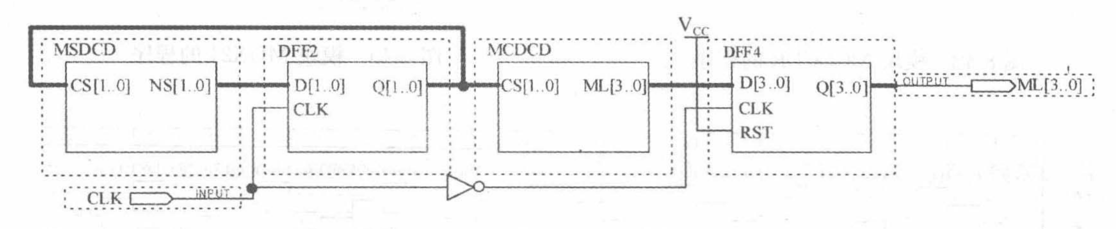


图 8-45 输出信号被 CLK 下降沿锁存的控制电路

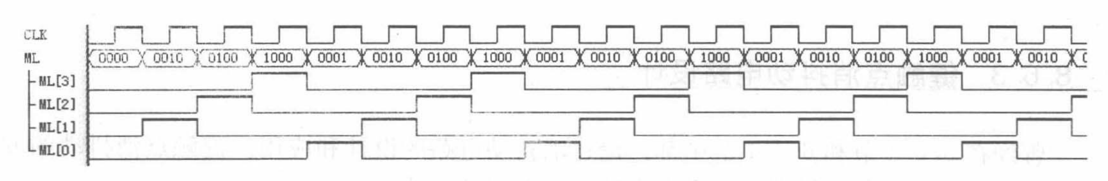


图 8-46 状态机输出被锁存后的时序仿真波形

3. 步进电机双向旋转控制电路设计

对图 8-45 电路稍作改进就能构成对步进电机实现双向旋转控制。电路如图 8-47 所示，其中的 S 是电机转向控制端。电路中的状态译码模块 MCDCDR 的程序如图 8-48 所示；模块 MCDCD 的程序即图 8-45 中同名模块的程序；模块 MUX21 是 4 位 2 选 1 多路选择器，对应的程序如图 8-49 所示。

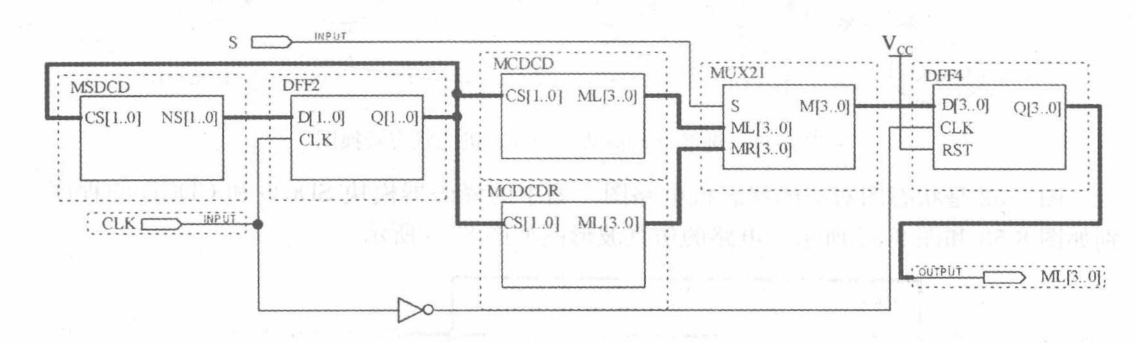


图 8-47 步进电机双向转动控制电路

图 8-50 是电路图 8-47 的时序仿真波形图。图中，当 S=1 和 S=0 分别对应的输出信号 ML 的波形序列是相反的，显然能控制电机不同的转向。

```
1  module  MCDCDR (CS,ML);
2   input [1:0] CS;
3   output [3:0] ML ; reg [3:0] ML ;
4   always @(CS)
5     case ( CS )
6       2'B00 :  ML<=4'B1000;
7       2'B01 :  ML<=4'B0100;
8       2'B10 :  ML<=4'B0010;
9       2'B11 :  ML<=4'B0001;
10      default :  ML<=4'B1000;
11    endcase
12  endmodule
```

图 8-48 模块 MCDCDR 的程序

```
1  module  MUX21 (S,ML,MR,M);
2   input S; input [3:0] ML,MR;
3   output [3:0] M; reg [3:0] M;
4   always @(S)
5    case ( S )
6      1'B0 :  M<=ML;
7      1'B1 :  M<=MR;
8      default :  M<=ML;
9    endcase
10  endmodule
```

图 8-49 模块 MUX21 的程序

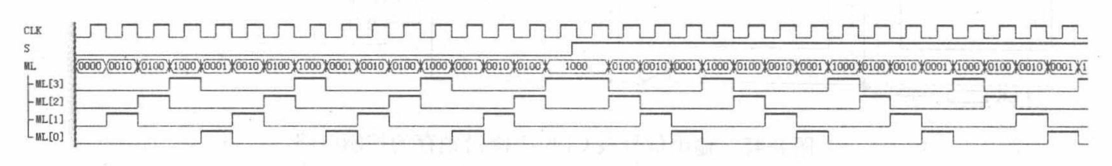


图 8-50 电路图 8-47 的时序仿真波形

8.6.3 键触点消抖动电路设计

曾经在 5.2.4 节和 6.6.1 节中都讨论过消抖动电路的设计和应用。键触点消抖动的方法可以有多种，在此不妨用状态机来设计一个消抖动电路。

图 8-51 描绘了一个键触点消抖动状态机的状态转换图。为了避开当键按下或松开时的随机抖动脉冲，共用了 6 个状态。用其中的 3 个状态测试键（设按下键后出现高电平）是否被按下，另 3 个状态则测试键是否被松开。状态图显示，只有当连续 3 个状态测出的键信号都是 1 时，才确定有键按下，使 KOUT 输出 1，否则返回原状态继续测试；之后，只有当连续 3 个状态测出的键信号都是 0 时，才确定键已松开，使 KOUT 输出 0。

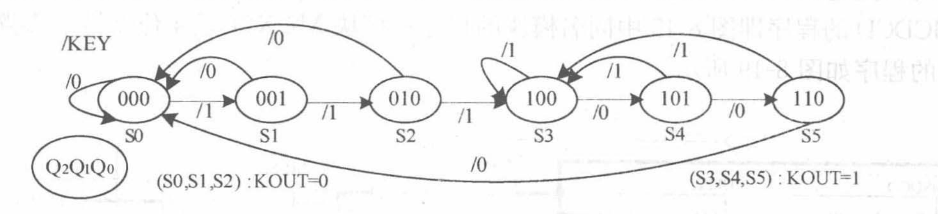


图 8-51 键触点消抖动电路状态机的状态转换图

图 8-52 是状态图对应的状态机电路图。图中的译码器模块 SDCD 和 CDCD 的程序分别如图 8-53 和图 8-54 所示。电路的仿真波形图如图 8-55 所示。

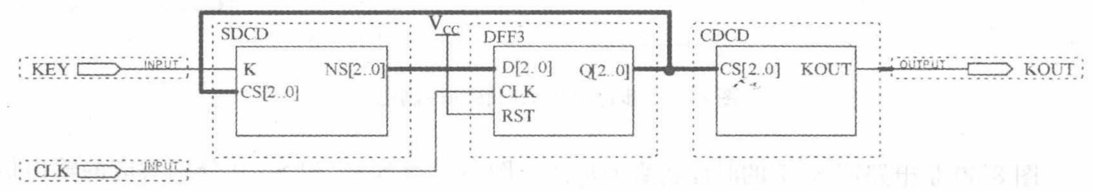


图 8-52 键触点消抖动电路原理图

```
1  module  SDCD (K,CS,NS);
2      input  K ; input [2:0] CS ;
3      output [2:0] NS ;
4        reg [2:0] NS ;
5    always @(K,CS)
6      case ( CS )
7       3'B000 : if (K==1'b1)  NS<=3'B001; else NS<=3'B000;
8       3'B001 : if (K==1'b1)  NS<=3'B010; else NS<=3'B000;
9       3'B010 : if (K==1'b1)  NS<=3'B011; else NS<=3'B000;
10      3'B011 : if (K==1'b0)  NS<=3'B100; else NS<=3'B011;
11      3'B100 : if (K==1'b0)  NS<=3'B101; else NS<=3'B011;
12      3'B101 : if (K==1'b0)  NS<=3'B000; else NS<=3'B011;
13      default :  NS<=3'B000;
14     endcase
15   endmodule
```

图 8-53 模块 SDCD 的程序

```
1  module  CDCD (CS,KOUT);
2    input [2:0] CS;
3    output KOUT;     reg KOUT ;
4    always @(CS)
5     case ( CS )
6         3'B000 :  KOUT <=1'B0;
7         3'B001 :  KOUT <=1'B0;
8         3'B010 :  KOUT <=1'B0;
9       default :  KOUT <=1'B1;
10     endcase
11    endmodule
```

图 8-54 模块 CDCD 的程序

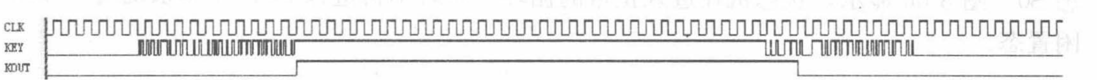


图 8-55 图 8-52 电路的仿真波形图

本节给出的状态机消抖动电路方案看上去稍嫌复杂，但实际上电路所耗用的硬件资源极省，若再增加两个状态进行键信号测试则会更省资源，逻辑综合后的资源利用报告显示，仅耗用了 3 个逻辑宏单元，其中包括 3 个 D 触发器。因此，这类过于简单的电路的消抖动效果不理想是预想中的事，在此只是用来说明状态机的一种设计和应用方法。更理想的方案是用计数器对按键和松键时的抖动脉冲进行计数，然后分析比较计数结果来确认是否是真实的键信号。此方案的描述和实际的硬件验证留给读者完成。

8.6.4 简易温控系统设计

设计一温控系统。要求在温度低于下限温度时打开加热开关；在高于上限温度时关闭加热开关，且在上限温度内打开紫外线灯 5 分钟进行空气消毒。为简化设计，测温系统可暂时不考虑。本例作以下假设：

- T1＝1：表示高于上限温度；T1＝0：表示低于上限温度。
- T2＝1：表示高于下限温度；T2＝0：表示低于下限温度。
- P＝1：表示打开加热电源开关；P＝0：表示关闭加热电源开关。
- R＝0：关闭并清 0 定时器，且关闭紫外线灯；R＝1：打开定时器，定时开始，同时打开紫外线灯。
- Q＝0：表示定时器处于关闭状态，或正在定时；Q＝1：表示定时时间到。

状态机的状态流程及状态编码分配如下：

- S0（000）：初始化恒温控制系统，包括使 P＝0、R＝0；下一状态进入 S1。这里的状态 0，即 S0 的状态编码定义为 000。
- S1（001）：使 P＝1，加热。下一状态进入 S2。
- S2（010）：测试上限温度标志 T1，若 T1＝1，下一状态进入 S3；若 T1＝0，下一状态仍然保持在状态 S2，等待加温。
- S3（011）：使 P＝0。启动定时器，打开紫外线灯，即使 R＝1；下一状态是 S4。

- S4（100）：测试定时信号，若 Q=1，下一状态进入 S5；若 Q=0，下一状态回到 S4。
- S5（101）：使 R=0，定时结束，关闭紫外线，下一状态进入 S6。
- S6（110）：测试下限温度标志 T2，若 T2=1，下一状态进入 S6；若 T2=0，下一状态进入 S1。
- S7（111）：下一状态进入 S0。

根据以上的条件假设及状态变化流程，可以做出此系统的状态图，如图 8-56 所示。状态图中的状态 S7 是一个闲置状态，为了确保状态机的自启动功能，电路将此闲置状态引入到初始态 S0。状态机在上电，及此后的系统复位（RST 低电平有效）后即进入初始态 S0。图 8-56 显示，状态机在进入正常的循环工作后不再进入 S0，除非系统复位或进入闲置态。

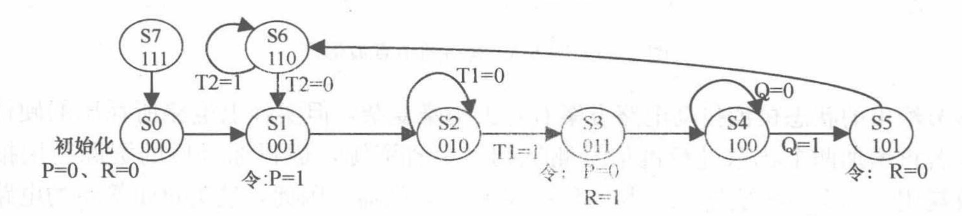


图 8-56　简易温控系统状态图

根据状态图和图 8-17 的 Moore 型有限状态机模型，可以做出此控制系统的电路图，如图 8-57 所示。图中的各模块和信号端都做了详细注释，在此不再作过多说明。

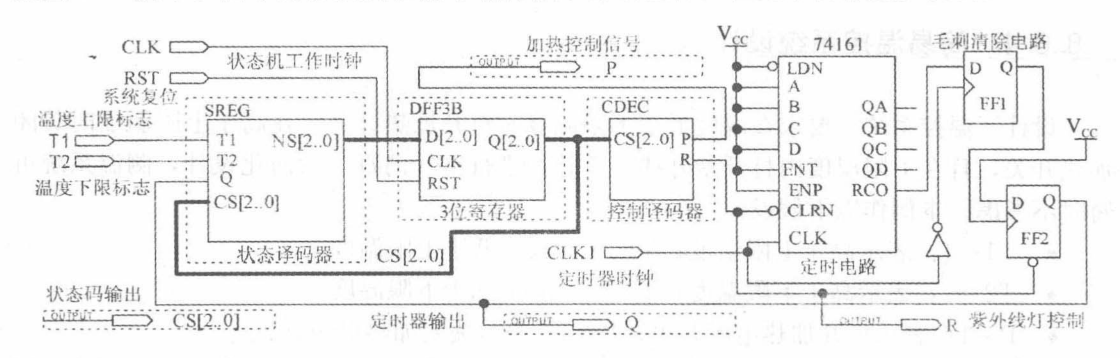


图 8-57　简易温控系统电路图

图中的状态译码器模块 SREG 和控制译码器模块 CDEC 的 case 语句表述分别如图 8-58 和图 8-59 所示。读者可根据各状态的功能和流程仔细核对这两个 case 代码的表述。

模块 DFF3B 是一个由 3 个 D 触发器构成的寄存器，结构与图 8-45 中的 DFF4 类似。图 8-57 中的其他电路构成了一个简易定时器。考虑到 74161 的计数进位输出 RCO 来自组合电路，有可能含有毛刺脉冲。为了避免误读，使用反相器和触发器 FF1 构成了一个毛刺去除电路（读者可以将 RCO 信号直接引出仿真，观察其毛刺情况）。

由于 RCO 的进位信号是一个窄脉冲，模块 SREG 有可能漏测这个信号，所以增加一个 D 触发器 FF2，用于锁存 RCO 信号。

```
1  module  SREG (T1,T2,Q,CS,NS);
2      input  T1,T2,Q ; input [2:0] CS ;
3      output [2:0] NS ;
4        reg [2:0] NS ;
5    always @(T1,T2,Q,CS)
6      case ( CS )
7       3'B000 : NS<=3'B001;
8       3'B001 : NS<=3'B010;
9       3'B010 : if (T1==1'b1)  NS<=3'B011; else NS<=3'B010;
10      3'B011 : NS<= 3'B100;
11      3'B100 : if (Q==1'b1)   NS<=3'B101; else NS<= 3'B100;
12      3'B101 : NS<=3'B110 ;
13      3'B110 : if (T2==1'b1)  NS<=3'B110; else NS<=3'B001;
14      3'B111 : NS<=3'B000;
15      default :  NS<=3'B000;
16     endcase
17   endmodule
```

图 8-58　状态译码器 SREG 的 case 语句表述

```
1  module  CDEC (CS,P,R);
2     input [2:0] CS;
3     output P,R;    reg P,R ;
4     always @(CS)
5       case ( CS )
6          3'B000 :  {P,R}<=2'B00;
7          3'B001 :  {P,R}<=2'B10;
8          3'B010 :  {P,R}<=2'B10;
9          3'B011 :  {P,R}<=2'B01;
10         3'B100 :  {P,R}<=2'B01;
11         3'B101 :  {P,R}<=2'B00;
12         3'B110 :  {P,R}<=2'B00;
13         3'B111 :  {P,R}<=2'B00;
14       default :  {P,R}<=2'B00;
15     endcase
16     assign  SOUT = CS ;
17    endmodule
```

图 8-59　控制译码器 CDEC 的 case 语句表述

一个能定时 5 分钟，或能指定定时时间的严格意义上的定时器的二进制位数应该比 74161 多得多，且定时器时钟 CLK1 的频率越高，定时精度也越高。在计数器位数和时钟频率一定的情况下，定时的时长由并行加载的数据决定。如果希望能随时更改定时的时长，则需将图 8-57 电路中的 74161 连接成可预置并行数据的结构。

图 8-60 是此状态机控制系统的仿真波形图，其中 CS 指示的是当前状态。仔细核对后可以发现，此图完全正确反应了状态机的所有功能要求。由于是属于对硬件系统的功能和时序特性的仿真，只要仿真结果正确，图 8-57 的电路一旦下载于 FPGA 后，能正常工作的可能性是极高的。

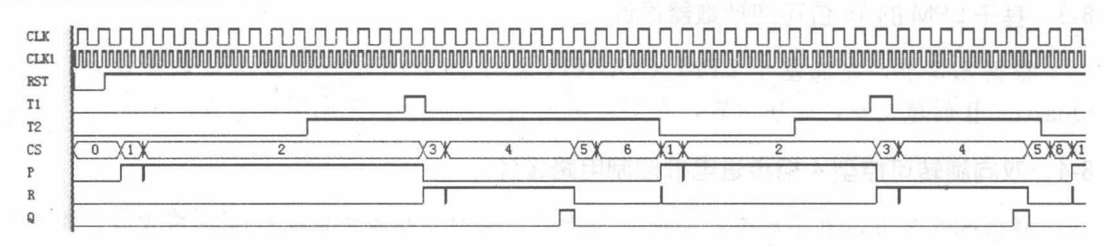


图 8-60　图 8-57 电路的时序仿真波形图

注意： 图 8-60 中，除了 CLK 和 CLK1 外，作为激励信号的 T1 和 T2 的电平是设计者自设的。此电平的恰当的设置对于是否能获得正确的仿真结果至关重要。



实　验

8-1　用 74 系列宏模块设计两种不同类型的计数器

(1) 根据 8.1.1 节，首先使用 74390 设计一个 2 位十进制计数器，然后使此计数器在新的工程中作为一个可调用的元件，用它构建一个 8 位十进制计数器。给出仿真结果，最后在 FPGA 上进行硬件验证。

(2) 根据 8.1.2 节，用 74161 模块设计一个十二进制加法计数器，并注意对计数器的可行性和可靠性考察；然后设计一个数控分频器。利用 Quartus Ⅱ 创建工程，绘制电路图，全程编译，时序仿真，并根据仿真波形作出说明，引脚锁定编译后下载 FPGA 中，在实验系统上硬件验证，完成实验报告。

8-2　基于一般模型的计数器设计

(1) 利用一般模型设计一个同步模7计数器，其状态图如图8-61所示。结合第6章介绍的Quartus Ⅱ流程来实现。

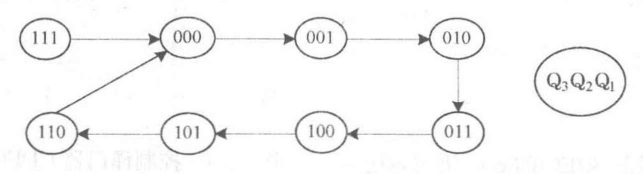


图8-61　实验8-2的状态图

(2) 基于自动设计方法的一般模型，设计一个模可控的同步加法计数器，要求当控制信号M=0时为六进制计数器，当M=1时为十二进制计数器。

(3) 设计一个基于一般模型的十进制加法计数器，利用Quartus Ⅱ创建工程，编辑电路图，时序仿真，并根据仿真波形作说明，在实验系统上硬件验证。注意计数器的自启动问题。

(4) 用自动设计技术完成基于计数器一般模型的可逆8位二进制计数器设计。

思考题1：叙述普通二进制计数器、一般模型、状态机之间的关系。

思考题2：在完成控制任务中，状态机中的状态译码器主要承担什么功能？

8-3　基于LPM的16位可逆计数器设计

根据8.5节，完成基于LPM_COUNTER的16位可逆可预置型计数器设计。利用Quartus Ⅱ创建工程，时序仿真，在实验系统上硬件验证。完成实验报告。

8-4　双向旋转可控型4相步进电机控制电路设计

根据8.6.2节，利用状态机完成双向旋转可控型4相步进电机控制电路设计。给出时序仿真和说明，最后在实验系统上进行硬件验证。完成实验报告。

8-5　键触点消抖动电路设计

根据8.6.3节，用状态机完成键触点消抖动电路的全部设计，对电路进行仿真。在硬件验证中参考实验7-2给出的方法。为了电路的可靠性，讨论状态机工作时钟频率的最佳值。

仍然基于状态机，探寻更好更可靠的设计方案，并验证之。

8-6　温控系统电路设计

(1) 根据8.6.4节，首先完成此节的设计内容，并验证仿真结果。针对系统中所需要的定时器，设计一个更灵活实用的定时器。

(2) 仿照实验任务(1)的设计，并查阅资料，用状态机设计一个电饭煲控制模型。要求对电饭煲工作的几个典型阶段，包括预约、升温、吸水、加热、糊化、焖饭和保温，配置实用定时器，完成温度控制系统设计。

8-7　序列发生器设计

用状态机设计一个序列发生器。设序列发生器可周期性输出编码1100100101，高位在前。

第9章 半导体存储器及其应用

半导体存储器是计算机和许多数字系统不可或缺的重要组成部分。本章首先介绍半导体存储器的基本概念、分类及其主要技术指标，以及各种存储器的基本结构、工作原理与存储器容量的扩展技术。为了能收到更好的学习效果，最后介绍两则利用 LPM 的 RAM 和 ROM 存储器的应用实例，并针对这些实例安排了实验。建议读者将学习的重点放在对存储器基本特性的了解和对存储器的实际使用技术的掌握上。

9.1 存储器概述

半导体存储器能存储各类数据、资料及计算机程序等大量信息，并且能够按照要求从相应的地址中取出。存储器种类繁多，适用面宽，除用于存储数据外，还可用于实现不同形式的逻辑函数和适用于许多特殊场合的逻辑功能，甚至能直接用于算术运算。

9.1.1 存储器分类

微电子技术的发展直接促进了半导体存储器技术的巨大进步，这一进步又推动了计算机技术的快速发展。现代的半导体存储器以其容量大、存取速度快、可靠性高、接口简单等特点，在计算机和数字系统中得到了广泛的应用。

存储器种类很多，除半导体存储器外还有许多其他类型的存储器。按照存储器的性质和特点来分类，存储器有不同的分类方法。

1. 按存储介质分类

按存储介质分类有半导体存储器、磁介质存储器、光存储器三大类。

2. 按存取功能分类

从存储功能上可分为只读存储器（Read-Only Memory，简称 ROM）和随机存取存储器（Random Access Memory，简称 RAM）两类，以下主要讨论这两类半导体存储器。

只读存储器在正常工作状态时，只能从中读取数据，而不能随时修改和写入数据。ROM 的优点是电路结构简单，数据一旦固化在存储器内部后，就可以长期保存，而且在断电后数据也不会丢失，属于数据非易失性存储器。在计算机中，只读存储器用于存放某些需要长期存放的信息，如数据、表格、专用程序等。

随机存取存储器与只读存储器的区别是，随机存储器 RAM 在正常工作状态时可随时向存储器里写入数据或从中读取数据。在存储器断电后信息就会全部丢失，因此 RAM 也称为易失性存储器。

3. 按制造工艺分类

根据制造工艺的不同，存储器可分为双极型存储器、MOS 型存储器等类型。双极型存储器以 TTL 触发器作为基本存储单元，具有速度快、价格高和功耗大等特点，主要用于高速应用场合，如计算机的高速缓存。MOS 型存储器是以 MOS 触发器或 MOS 电路为存储单元，具有工艺简单、集成度高、功耗小、价格低等特点，目前计算机中的大容量内存存储器主要采用 MOS 型存储器。

4. 根据数据的输入/输出方式分类

存储器可分为串行存储器和并行存储器。串行存储器中数据输入或输出采用串行方式，并行存储器中数据输入或输出采用并行方式。显然，并行存储器读写速度快，但数据线和地址线占用芯片的引脚数较多，且存储容量越大，所用引脚数目越多。串行存储器的速度比并行存储器慢一些，但芯片的引脚数目少了许多。

9.1.2 半导体存储器的技术指标

存储器的性能指标有很多，例如存储容量、存取速度、封装形式、电源电压、功耗等。但就实际应用而言，最重要的性能指标是存储器的存储容量和存取时间。下面就这两项性能指标的具体情况给予说明。

1. 存储容量

存储容量是指存储器所能容纳的二进制数据的总量。存储器容量的计算公式为：字数（m）×字长（n）。存储器的容量通常按照二进制的位（bit）或字节（Byte）来计算，其中 1Byte＝8bit。

存储器容量通常用 K、M 或 G 字节来表示。其中 $1K=2^{10}=1024$，$1M=2^{20}=1024K$，$1G=2^{30}=1024M$。

2. 存取速度

存储器的存取速度可用“存取时间”和“存储周期”这两个时间参数来衡量。

存取时间（Access Time）是指从微处理器发出有效存储器地址，从而启动一次存储器读/写操作到该操作完成所经历的时间。显然存取时间越短，则存取速度越快。目前，高速缓冲存储器的存取时间已小于 20ns；中速存储器则在 60～100ns 之间；低速存储器在 100ns 以上。

存储周期（Memory Cycle）是连续启动两次独立的存储器操作所需的最小时间间隔。由于存储器在完成读/写操作之后需要一段恢复时间，所以存储器的存储周期略大于存储

器的存取时间。如果在小于存储周期的时间内连续启动两次或两次以上存储器访问，那么存取结果的正确性将不能得到保证。

9.2 随机存取存储器

随机存储器可分为静态 RAM 和动态 RAM。在此主要介绍静态 RAM 的结构、存储单元的组成和动态 RAM 存储单元的基本概念。不同类型的 RAM 也有许多共性，如可读可写，非破坏性读出，写入时覆盖原内容；可随机存取，存取任一单元所需的时间相同；易失性（或挥发性），即当断电后，存储器中的内容立即消失。

9.2.1 RAM 的分类及其结构

RAM 的类型繁多，结构各异，对应不同的适用范围和应用领域。这些都必须根据实际需要来决定使用什么类型的 RAM。

1. RAM 分类

随机存储器也称为可读/写存储器，简称 RAM。根据制造工艺的不同，可分为双极型和 MOS 型存储器。双极型存储器由于集成度低，功耗大，目前常用的 RAM 主要是 MOS 型的。根据工作原理的不同，又可分为静态 RAM（Static RAM，简称 SRAM）和动态 RAM（Dynamic RAM，简称 DRAM）。

从结构上看，静态 RAM 使用触发器作为存储元件，因此只要电源维持不变，就可保持数据不被丢失。SRAM 的缺点是占用硬件资源多，难以构成大规模器件。

DRAM 则以 MOS 管的栅极电容作为存储单元。DRAM 电路简单，集成度高，但由于栅极电容量非常小，且存在漏电流情况，存储在电容上的电荷会很快泄漏，致使信号保存时间短暂。为防止信号丢失，动态 RAM 必须配备刷新电路，给栅极电容补充电荷。即对电容上的数据定时刷新，确保维持原有数据，否则就不能长期保存数据。

当电源被撤除后，SRAM 和 DRAM 存储的数据都会丢失，因此都属于易失性存储器。SRAM 和 DRAM 可以进一步分为更多的类型，其分类结构如下所示：



从时序上分，SRAM 有两类，即同步型和异步型；从接口方式上分，SRAM 有单口、双口和多口类型。动态 RAM 按制造工艺的不同，可分为动态随机存储器（Dynamic

RAM）、扩展数据输出随机存储器（Extended Data Out RAM）和同步动态随机存储器（Synchromized Dynamic RAM，简称 SDRAM）。

常用的 DRAM 有以下几种类型：

- SDRAM，它在一个 CPU（计算机的中央处理器）时钟周期内可完成数据的访问和刷新，即可与 CPU 的时钟同步工作。SDRAM 的工作频率目前可达 150MHz，存取时间约为 5～10ns，最大数据率为 150MB/s，是当前微型计算机中流行的标准内存类型。
- RDRAM（Rambus DRAM），是由 Rambus 公司开发的高速 DRAM。其最大数据率可达 1.6GB/s。
- DDR DRAM（Double Data Rate DRAM），是 SDRAM 的改进型。在时钟的上升沿和下降沿都可以传送数据，数据率可达 200～800 MB/s。主要用在主板和高速显示卡上。

不同类型的 RAM 在工作中却有许多共性，如：可读可写，即非破坏性读出，写入时覆盖原内容；随机存取，即存取任一单元所需的时间相同；易失性（或挥发性），即当断电后，存储器中的内容立即消失；等等。

2. RAM 的基本结构

RAM 的电路结构如图 9-1 所示，主要由地址译码器、存储矩阵、读/写控制电路三部分组成。每一个存储单元存放一位二进制数据。为了存取方便，通常将存储单元设计成矩阵形式。存储矩阵由许多结构相同的基本存储单元排列组成，而每一个基本存储单元可以存储一位二进制数据；在地址译码器和读写控制电路的作用下，将存储矩阵中指定的存储单元中的数据读出或将数据写入指定的存储单元中。

地址译码器的作用是将输入的地址译成高（或低）电平输出信号，使指定的地址单元与读/写控制电路接通，并在读/写信号控制下进行读/写操作。

通常有两种地址译码方式：单译码和双译码。在大容量存储器中常采用双译码，这种译码方法是将地址分为行地址和列地址两部分，分别对行地址和列地址进行译码，由它们共同选择存储矩阵中欲读/写的存储单元。

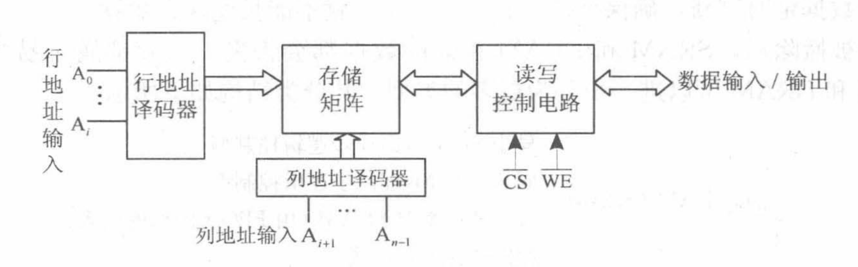


图 9-1　RAM 的电路结构框图

$\overline{WE}$是读/写控制信号；$\overline{CS}$是片选控制输入端，低电平有效。数字系统一般由许多芯片组成，而系统每次只选中其中一片进行读/写操作。对$\overline{CS}$端控制的作用就是用于对 RAM 芯片进行选择。系统中扩展多片 RAM 后，每片 RAM 上均有片选信号$\overline{CS}$，当$\overline{CS}=0$时，此 RAM 芯片被选中；而取$\overline{CS}=1$的其他 RAM 芯片无效。其数据输入/输出（I/O）

端口呈现高阻态，效果上与数据总线呈脱离状态。

读/写控制电路的作用是对存储器的工作状态进行控制。当$\overline{CS}=0$时，RAM为正常工作状态；若$\overline{WE}=1$，则执行读操作，存储单元里的数据将送到输入/输出口上；若$\overline{WE}=0$，则执行写操作，加到输入/输出口的数据被写入存储单元中；而当$\overline{CS}=1$时，RAM的输入/输出端呈高阻态，这时就无法对RAM进行读/写操作了。

9.2.2 SRAM的结构

为了适应不同的应用场合，SRAM有不同的结构类型和不同的技术指标，并且随着集成电路技术的发展，还会出现更多类型的产品。但它们的基本结构和工作原理则大同小异，以下仅介绍SRAM典型的单元结构。

1. SRAM的基本存储单元

静态RAM的基本存储单元通常由电平触发型D触发器（或类似的时序单元）加上一些控制电路所组成。图9-2所示的是一个SRAM的单元存储结构简图。图中SELECT是选通控制端，$\overline{READ}$是读/写控制端，DATA in/out是数据输入/输出端。

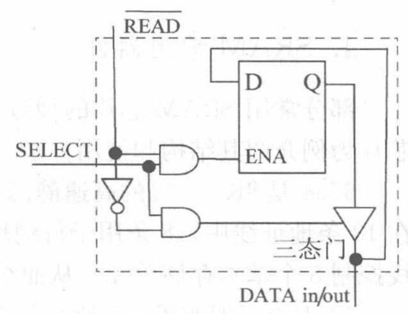


图9-2 静态RAM基本存储单元

当SELECT为低电平时，三态门处于断开状态，DATA对外呈现高阻态，禁止对D触发器读/写操作。当SELECT和$\overline{READ}$同时为高电平时，对存储单元进行写操作。数据从DATA端送到触发器的D端；当SELECT为高电平、$\overline{READ}$为低电平时，就将数据保存到D触发器中，同时能对存储单元进行读操作。此时三态门打开，Q端的数据经三态门输出。图9-3是由图9-2的SRAM单元构成的一个RAM存储器模型。

2. 用D触发器构成SRAM结构

图9-3是用D触发器组成的4×2SRAM存储器单元结构，它构成了4个2位（bit）的存储字。图中的主要元件有2-4译码器，8个1位的基本存储单元和输入输出电路。

WE是写允许信号；A_1、A_0是地址信号，经译码器译码后产生存储单元选择信号；D_1、D_0是存储器的数据输入端；Q_1、Q_0是存储器的数据输出端。

3. SRAM存储矩阵结构

SRAM中的基本存储单元以行和列组织起来，图9-4是一个n×4阵列的基本SRAM阵列。行中所有基本存储单元共享相同的行选择线。数据线进入给定列中的每个单元内，并经过数据输入/输出缓冲器和控制电路，成为单个数据线上的一个输入或输出数据。

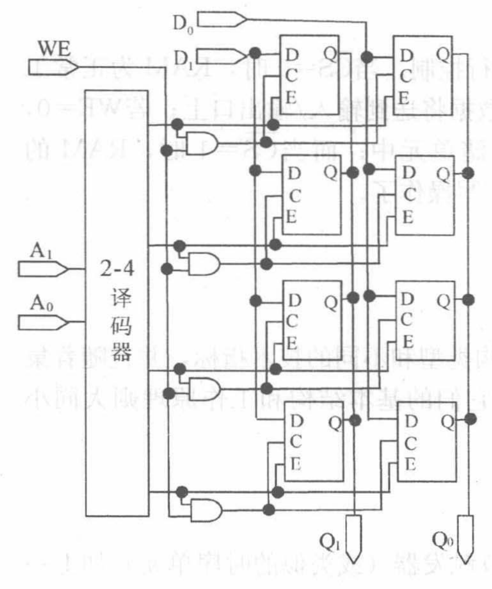


图 9-3 4×2 静态 RAM 结构

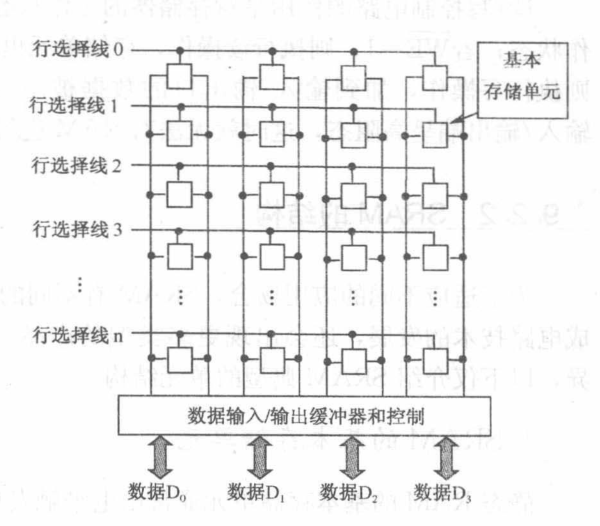


图 9-4 基本 SRAM 阵列

4. SRAM 常用器件

部分常用 SRAM 芯片的型号有 6116、6264、62256、628128 等。这里以 SRAM 6264 芯片为例介绍其结构和使用方法。

6264 是 8K×8 位的高速静态 CMOS 可读写存储器，片内共有 65536 个基本存储单元。在 13 条地址线中，8 条用于行地址译码输入，5 条用于列地址译码输入，每条列地址译码线控制 8 个基本存储单元，从而组成了 256×256 的存储单元矩阵。

6264 的引脚如图 9-5 所示。在 28 个引脚中有 13 条地址线（$A_0 \sim A_{12}$）、8 条数据线（$I/O_0 \sim I/O_7$）、1 条电源线（Vcc）和 1 条地线（GND）。此外还有 3 条控制线，即片选线 $\overline{CE_1}$ 和 CE_2、输出允许 $\overline{OE}$、写允许 $\overline{WE}$。其中 $\overline{CS_1}$、CS_2、$\overline{OE}$ 和 $\overline{WE}$ 的组合决定了 6264 的工作方式，详细情况如表 9-1 所示。

引脚	序号	序号	引脚
NC	1	28	Vcc
A_{12}	2	27	$\overline{WE}$
A_7	3	26	CS_2
A_6	4	25	A_8
A_5	5	24	A_9
A_4	6	23	A_{11}
A_3	7	22	$\overline{OE}$
A_2	8	21	A_{10}
A_1	9	20	$\overline{CS_1}$
A_0	10	19	I/O_7
I/O_0	11	18	I/O_6
I/O_1	12	17	I/O_5
I/O_2	13	16	I/O_4
Vss	14	15	I/O_3

6264

图 9-5 6264 芯片引脚图

表 9-1 6264 的工作方式

CS_2	$\overline{CS_1}$	$\overline{OE}$	$\overline{WE}$	工作方式
1	0	0	1	读
1	0	1	0	写
0	1	×	×	未选

9.2.3 DRAM 工作原理

DRAM 在微型计算机中应用非常广泛，如微机中的内存条（主存）、显卡上的显示存储器等，几乎都使用了 DRAM。前面曾提到 DRAM 是靠 MOS 电路中的栅极电容来存储信息的。由于电容上的电荷会逐渐泄漏，需要定时充电以维持存储内容不丢失，这称为动态刷新。为此 DRAM 需要设置刷新电路，刷新定时的间隔一般为几微秒至几毫秒。所以相应的外围电路就比 SRAM 复杂许多。

DRAM 的优势是集成度高（存储容量大，可达 1Gb/片以上）、功耗低。与静态 RAM 一样，动态 RAM 也是由许多基本存储单元按行、列形式构成的二维存储矩阵。在基本存储单元电路中，二进制信息是保存在 MOS 管栅极电容上的。若电容上充有电荷，则表示存有信息“1”；电容上无电荷时表示所存信息为“0”。即动态 RAM 是利用电容存储电荷的原理来保存信息的。

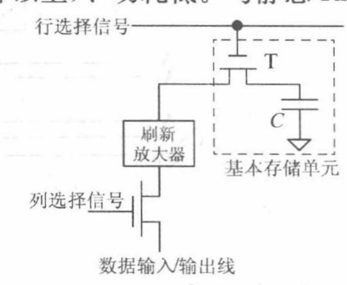


图 9-6 单管 DRAM 基本存储单元电路

目前，动态 RAM 基本存储单元是由一个 MOS 管和一个小电容构成的，故称为“单管动态 RAM 基本存储单元电路”，其结构如图 9-6 所示。

对单管动态 RAM 存储电路进行读操作时，通过“行地址译码器”使某一条行选择线为高电平，则该行上所有基本存储单元中的 MOS 管 T 导通。这样，各列上的刷新放大器便可读取相应电容上的电压值。刷新放大器灵敏度很高，可将电容上的电压放大后转换为逻辑“1”或“0”，并控制将其重写到存储电容上。“列地址译码器”电路产生列选择信号，使选中行和该列上的单管动态 RAM 存储电路受到驱动，从而输出数据。在进行写操作时，被行选择信号、列选择信号所选中的单管动态 RAM 存储电路的 MOS 管 T 导通，通过刷新放大器和 T 管，外部数据输入/输出线上的数据被送到电容 C 上保存。

由于任何电容均存在漏电效应，所以经过一段时间后电容上的电荷会流失殆尽，所存信息也就丢失了。尽管每进行一次读/写操作，实际上是对单管动态存储电路信息的一次恢复或增强，但是读/写操作的随机性不可能保证在一定时间内内存中所有的动态 RAM 基本存储单元都会有读/写操作。

对电容漏电而引起信息丢失这个问题的解决办法是定期地对内存中所有动态 RAM 存储单元进行刷新（Refresh），使原来表示逻辑“1”电容上的电荷得到补充，而原来表示逻辑“0”的电容仍保持无电荷状态。所以刷新操作并不改变存储单元的原存内容，而是使其能够继续保持原来的信息存储状态。

9.2.4 SRAM 的扩展方法

由于单片 RAM 存储器的容量有限，当不能满足系统对存储容量的要求时，可以进行扩展，这包括位扩展和字扩展。其方法是将若干片 RAM 组合起来。

1. 位扩展

当一个 RAM 芯片的每个字的位数（字长）小于系统要求时，可以进行位扩展。位扩展的方法是将各片 RAM 的地址线、读写控制线 $R/\overline{W}$、片选线$\overline{CS}$共用，数据线并行。如将两片 8K×8 位的 SRAM 扩展为 8K×16 位 SRAM，其连接图如 9-7 所示。

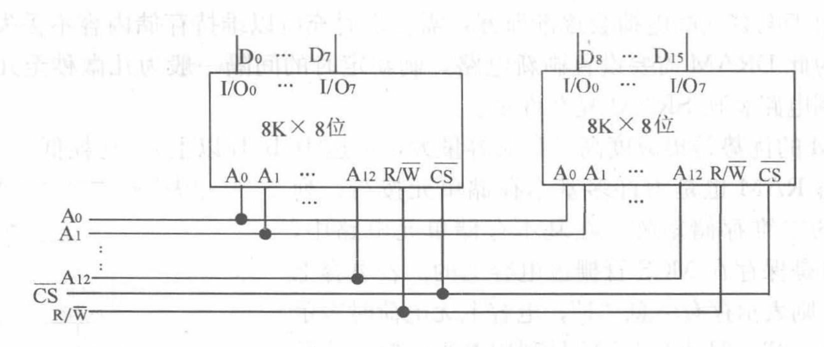


图 9-7 RAM 的位扩展法

2. 字扩展

当一片 RAM 芯片的字数小于系统要求时，就需要进行字扩展。字扩展时，导致地址线增加，每增加一位地址，可寻址单元数就增加一倍。字扩展的方法是将数据线、读/写控制线 $R/\overline{W}$、地址线分别并联，用高位地址经过译码后产生不同状态，以分别控制各片的$\overline{CS}$。如将两片 8K×8 位扩展为 16K×8 位 RAM，线路连接图如图 9-8 所示。其中的 A_{13} 通过一个反相器分别控制选通左右两个 RAM。

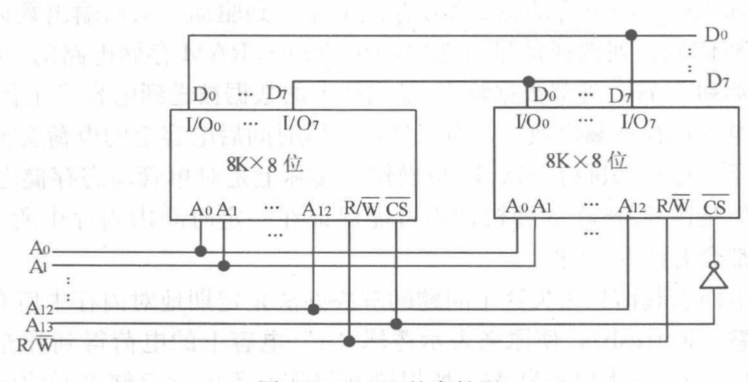


图 9-8 RAM 的字扩展

对于芯片的字数和位数都小于系统要求的情况，必须同时进行字扩展和位扩展。

9.3 只读存储器

按照数据写入方式的不同，只读存储器可分为掩膜 ROM、一次可编程 ROM（OTP

ROM)、可擦除可编程 ROM（EPROM)、电擦除可编程只读存储器（E^2PROM)，以及数据可在线写入和非易失性保存的 FLASH 存储器等。ROM 的优点是电路结构简单，数据一旦固化在存储器内部后，就可以长期保存，而且在断电后数据不会丢失，属于所谓非易失性存储器。在计算机中，只读存储器用于存放某些需要长期存放的固定信息，如数据、表格、专用程序等。

本节介绍常用的只读存储器 ROM 以及可编程 ROM 的结构与工作原理。

9.3.1 ROM 分类与结构

为了适应不同的应用场合，与 RAM 一样，ROM 也有多种类型。这里简要介绍 ROM 的类型和基本结构。

1. ROM 的分类

ROM 是存储固定信息的存储器。与 RAM 不同，ROM 中的信息是由专用装置预先写入的，在正常工作过程中只能读出不能写入。ROM 的用途是用来存放不需要经常修改的程序或数据，如计算机系统中控制启动和初始化的 BIOS 程序、系统监控程序、显示器字符发生器中的点阵代码等。

ROM 从功能和工艺上可分为掩膜 ROM、可编程的 PROM、EPROM 和 EEPROM 等多种类型，ROM 的分类情况大致如下：

ROM
- 掩膜 ROM
- PROM （可编程 ROM)
- EPROM （可擦除 PROM)
- UV EPROM （紫外线擦除 EPROM)
- EEPROM （电可擦除 EPROM)
  - 并行 EEPROM
  - 串行 EEPROM
- 嵌入式可编辑 ROM（由 FPGA 内部的嵌入式存储器编辑而成的 ROM)
- OTP ROM （一次性可编程 ROM)
- FLASH ROM （闪存)

2. ROM 的结构

ROM 的电路结构如图 9-9 所示，由存储矩阵、地址译码器和输出控制电路三部分组成。存储矩阵由许多基本存储单元排列而成。基本存储单元可以由二极管构成，也可以由

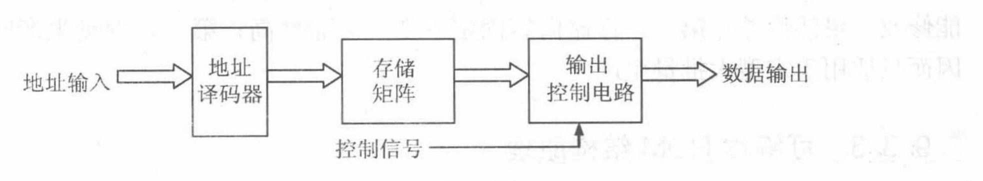


图 9-9 ROM 的电路结构框图

双极型三极管或MOS管构成。每个基本存储单元能存放一位二进制数据，每一个或一组基本存储单元有一个对应的地址。

地址译码器的作用是将输入的地址译成相应的控制信号，利用这个控制信号从存储矩阵中选出指定的单元，将其中的数据从数据输出端输出。

输出控制电路通常由三态输出缓冲器构成，其作用有两个：一是提高存储器的驱动能力，二是实现输出三态控制，以便与系统的总线连接。

9.3.2 掩膜 ROM

掩膜式ROM通常采用MOS工艺制作。在芯片制造厂家生产时，根据用户提供的需要写入ROM的数据或程序，即采用二次光刻板的图形，即掩膜，而将其直接写入（固化）的，因此称为掩膜ROM。掩膜ROM中的内容制成后，用户便不能修改，只能读出。

图9-10所示是一个简单的4×4位的MOS型ROM存储矩阵模型，采用单向（横向）译码结构，2位地址线 A_1A_0 译码后产生的4个输出，分别对应4条横向的字线（W_0～W_3）。以此可分别选中4个ROM存储单元之一，每个存储单元有4位，分别对应于4条纵向的位线（D_3～D_0）。当此掩膜ROM已被写入数据后，其结构内的字线和位线的交叉处有的连有MOS管，有的没有连接MOS管。连有MOS管的位，读出为“0”；没有连接MOS管读出为“1”。这是因为矩阵中的MOS管在工艺上可击穿接地，而最上缘的4个MOS管相当于4只上拉电阻。例如，若输入地址（A_1A_0）=10，译码后选中“字线2”，于是，根据图9-10，输出的数据（D_3～D_0）=1010。

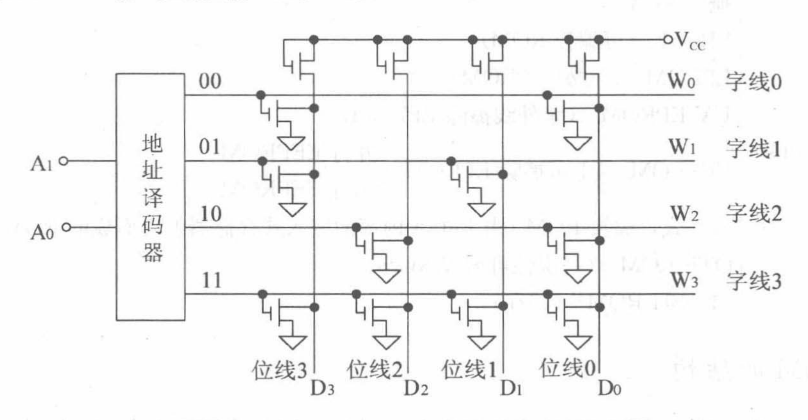


图9-10 4×4位的MOS型ROM存储矩阵

掩膜式ROM的主要特点是：第一，存储的内容由制造厂家一次性写入，写入后便不能修改，灵活性差；第二，存储内容固定不变，可靠性高；第三，少量生产时造价较高，因而只适用于定型大批量生产。

9.3.3 可编程 ROM 结构原理

可编程ROM便于用户根据自己的需要来写入特定的信息。通常，厂家生产的可编程

ROM事先并未存入任何程序和数据。存储矩阵的所有行、列交叉处均连接有二极管、三极管或MOS管。可编程ROM出厂后，用户可以利用芯片的外部引脚输入地址，对存储矩阵中的二极管、三极管或MOS管进行选择，使其写入特定的二进制数据。

根据存储矩阵中存储单元电路的结构不同，可编程的ROM有PROM、EPROM和EEPROM等三种。

1. 可编程ROM

曾在4.9.2节中介绍过PROM（Programmable ROM，简称PROM）内部的结构，当时是将其作为PLD器件来考察的，而现在则将其作为数据存储器来讨论，作用不同，但结构是相同的。

可编程PROM与掩模ROM的不同处是芯片在出厂时，前者所有的存储单元均被加工成同一状态“0”或“1”，用户可根据需要通过编程器将某些存储单元的状态变成另一状态“1”或“0”。但这种编程只能进行一次，一旦编程完毕，其内容便不能改写。

PROM的存储单元通常有两种电路形式：一种是由二极管构成的结击穿型电路；另一种是由晶体三极管组成的熔丝烧断型电路，其结构示意图如图9-11所示。

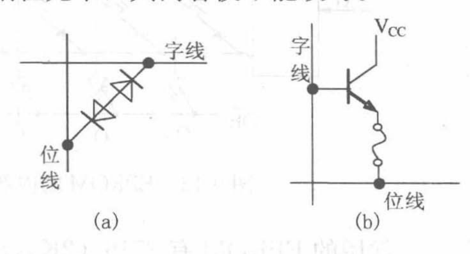


图9-11 击穿型和熔丝型存储单元电路

击穿型PROM中，每个存储单元都有两个背靠背的二极管，如图9-11（a）所示。这两个二极管将字线和位线断开，相当于每个存储单元都存入了“0”。用户在编程时，可根据需要对选中的存储单元加上一个高电压和大电流，将其反向二极管击穿，仅剩下一个正向导通的二极管，这时位线和字线接通，该存储单元相当于存有信息“1”。因此，这种编程是一次性的。

熔丝型PROM中，每个存储单元都有一个带熔丝的晶体三极管，其连接图如图9-11（b）所示。用户编程是逐字逐位进行的，根据需要写入的信息，按字线和位线选择某个存储单元。通过施加规定宽度和幅度的脉冲电流，将该连接三极管发射极的熔丝熔断，使该存储单元的状态被改变成与原状态相反的状态。熔丝熔断后，便不可恢复，显然，编程也是一次性的。

PROM编程虽是由用户而不是生产厂家完成的，增加了灵活性，但编程是一次性的，目前已很少使用了。

2. 可擦除可编程ROM

EPROM（Erasable PROM）作为一种可以多次擦除和重写的ROM，其内部结构如图9-12所示。它克服了掩膜式ROM和PROM只能一次性写入的缺点，满足了实际工作中需要多次修改程序或数据的可能，使得EPROM得到了广泛的应用。使用EPROM的前提条件是必须将存储矩阵中原有的程序或数据擦除。

通常，EPROM芯片上方有一个石英玻璃窗口，当用一定波长（如2537Å）、一定光强（如12000μW/cm^2）的紫外线透过窗口照射时，所有存储电路中浮栅上的电荷会形成

光电流泄放掉，使浮栅恢复初态。一般照射20～30分钟后，读出各单元的内容均为FFH，即所有单元的数据都为“1”，说明EPROM中内容已被擦除。

EPROM的擦除和编程写入是采用专门的编程设备完成的。因此，对于编程好的EPROM要用不透光的胶纸将受光窗口封住，以保护芯片不受荧光或太阳光的紫外光照射而造成信息丢失。图9-13是紫外线擦除型EPROM 2764（UV EPROM）的外观图。

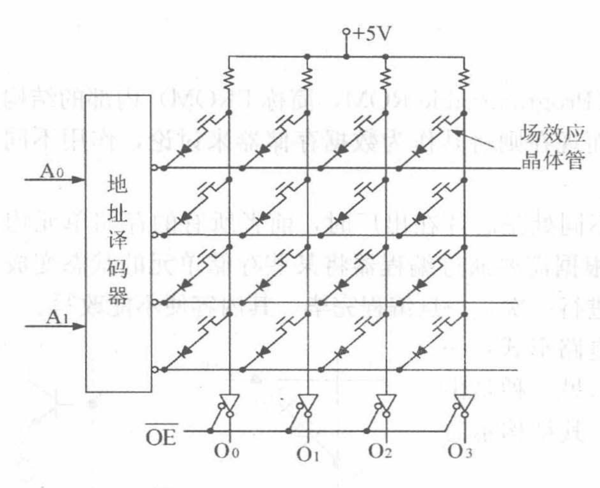


图9-12 EPROM的内部结构

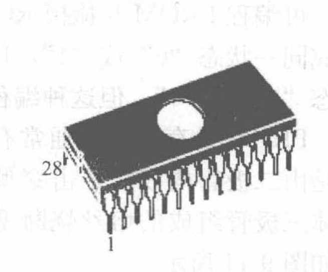


图9-13 EPROM 2764的外观图

常用的EPROM有2716（2K×8位）、2732（4K×8位）、2764（8K×8位）和27512（64K×8位）等。图9-14所示是容量为8K×8位的2764的引脚图。图中，V_{PP}为编程电源，$\overline{CE}$为片选信号，$\overline{PGM}$为编程脉冲信号，$\overline{OE}$为输出允许信号，A_0～A_{12}为地址信号，Q_0～Q_7为数据信号。对EPROM的编程和读写控制模式如表9-2所示。

引脚	序号	序号	引脚
V_{PP}	1	28	V_{CC}
A_{12}	2	27	$\overline{PGM}$
A_7	3	26	NC
A_6	4	25	A_8
A_5	5	24	A_9
A_4	6	23	A_{11}
A_3	7	22	$\overline{OE}$
A_2	8	21	A_{10}
A_1	9	20	$\overline{CE}$
A_0	10	19	Q_7
Q_0	11	18	Q_6
Q_1	12	17	Q_5
Q_2	13	16	Q_4
V_{SS}	14	15	Q_3

2764

图9-14 EPROM 2764的外部引脚图

目前市场上也有一些EPROM芯片没有设玻璃窗口的，由于成本低，没有准备重复使用，所以只能当一次性PROM用。

表9-2 EPROM的操作模式（V_P＝12V±0.5%）

操作模式	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	V_{PP}	Q_0～Q_7
读数据	V_{IL}	V_{IL}	V_{IH}	V_{CC}	数据输出
禁止输出	V_{IL}	V_{IH}	V_{IH}	V_{CC}	高阻
编程	V_{IL}	V_{IH}	V_{IL}脉冲	V_P	数据输入
校验	V_{IL}	V_{IL}	V_{IH}	V_{CC}	数据输出

3. 电可擦型可编程ROM

EPROM虽然可以多次编程，具有较好的灵活性，但在对它编程时，必须将芯片从电路板上拆下来，利用紫外线光源擦除后重写，因而给实际应用带来不便，并且此类器件的重复擦写次数也不多。

电可擦除型可编程只读存储器 EEPROM 也称 E^2PROM（Electrically Erasable PROM）。与 EPROM 擦除时把整个芯片的内容全变成“1”不同，EEPROM 的擦除可以按字节分别进行。字节的编程和擦除都只需短得多的时间，约 5～10ms，并且多数无需将芯片从电路板上拔下，以及用诸如紫外线光源照射等特殊操作，因此可以在系统（在系统处于工作条件下）进行擦除和编程写入。重复擦写次数约为 EPROM 的数十倍。

为了编程和擦除的方便，有些 EEPROM 芯片把其内部存储器分页（或分块），可以按字节擦除、按页擦除或整片擦除，对不需要擦除的部分可以保留。

常见的 EEPROM 芯片有 28C16、28C17、28C64、28C256 等。图 9-15 所示的是容量为 8K×8 位的 2864 芯片的引脚图。图中，$\overline{CE}$为片选信号，$\overline{WE}$为写控制信号，$\overline{OE}$为输出允许信号，A_0～A_{12}为地址信号，D_0～D_7 为数据信号。

【例 9-1】 将 2 片 8K×8 位的 EPROM2764 扩展成 8K×16 位的存储器。

解：单片 EPROM 芯片 2764 的存储容量为 8K×8 位，即字长为 8 位。若需要字长为 16 位的存储器 8K×16 位，则可将两片 2764 按照图 9-16 所示的位扩展方式进行容量扩展。图 9-16 中，两片 2764 的 13 位地址线（A_{12}～A_0）、片选信号$\overline{CE}$和输出允许信号$\overline{OE}$都公用。当$\overline{CE}$=0、$\overline{OE}$=0 时，来自 13 位地址总线的每个地址同时选中两片 2764 中各一个存储单元，读出两个具有相同地址的存储单元的 8 位二进制数值，即输出 16 位数值。

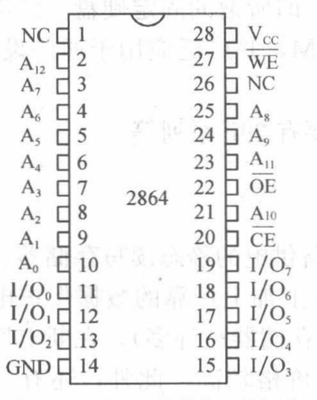


图 9-15 2864 的外部引脚图

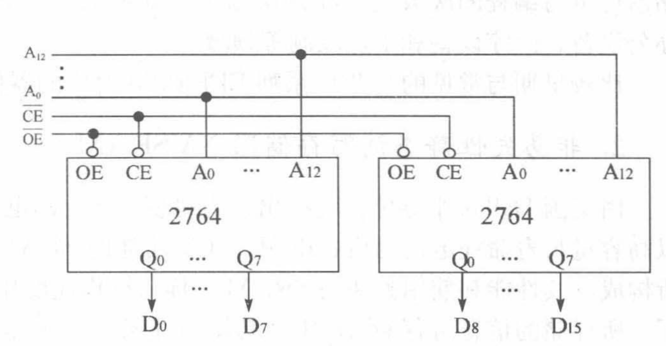


图 9-16 两片 2764 扩展成 8K×16 位 EPROM 组

【例 9-2】 对 EPROM 进行字扩展，将 8 片 2764 扩展成 64K×8 位的程序存储器。

解：如图 9-17 所示，8 片 2764 的低 13 位地址线（A_{12}～A_0）和输出允许信号$\overline{OE}$公用

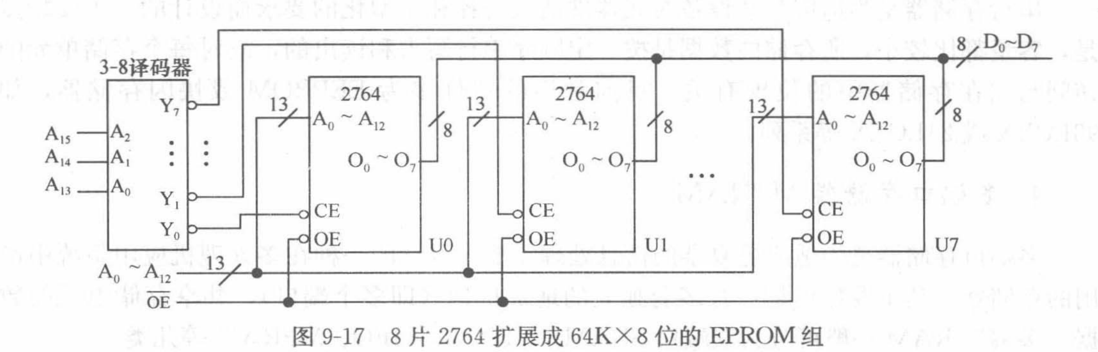


图 9-17 8 片 2764 扩展成 64K×8 位的 EPROM 组

信号，8 片 2764 的片选信号$\overline{CE}$则是由高 3 位地址（A_{15}～A_{13}）通过 74LS138 译码获得。（A_{15}～A_{13}）译码出的每个不同取值可选中一片 2764。例如当（A_{15}～A_{13}）=000 时，译码器输出端 Y_0 为 0，Y_1～Y_7 为 1，图中 U0 被选中，其他的 2764 则被禁止工作。此时，U0 中 8KB 存储单元的地址范围是：0000H～1FFFH。

类似地，通过分析可得到其他 2764 的地址范围，U1：（2000H～3FFFH），U2：（4000H～5FFFH），U3：（6000H～7FFFH），U4：（8000H～9FFFH），U5：（A000H～BFFFH），U6：（C000H～DFFFH），U7：（E000H～FFFFH）。

9.3.4 其他类型的存储器

随着集成电路制造工艺迅速发展，出现了适用于各种不同用途、不同环境和不同需求的新型的存储器。

1. 快闪存储器 Flash Memory

采用与 EPROM 中的叠栅 MOS 管相似的结构，同时保留了 EEPROM 用隧道效应擦除的快捷特性。理论上属于 ROM 型存储器，但功能上却相当于 RAM。单片容量已达数千 MB，并正在推出容量更大、存储速度更高的快闪存储器，即所谓的固定硬盘。此类存储器可重写编程的次数达 100 万次以上，基本取代了 EPROM，并广泛应用于通信设备、办公设备、医疗设备和工业控制等领域。

比较早期与常见的与 28C 系列 EEPROM 对应的器件主要有 29F 系列等。

2. 非易失性静态读写存储器 NVSRAM

由美国 Dallas 半导体公司推出，为封装一体化的电池后备供电的静态读写存储器。它以高容量长寿命锂电池为后备电源，在低功耗的 SRAM 芯片上加上可靠的数据保护电路所构成。其性能和使用方法与 SRAM 一样（操作速度比快闪存储器高许多），在断电情况下，所存储的信息可保存 10 年。其缺点主要是体积稍大，价格较高。此外，还有一种 NVRAM，不需电池作后备电源，它的非易失性是由其内部机理决定的。

3. 串行存储器

串行存储器是为适应某些设备对元器件的低功耗和小型化的要求而设计的。主要特点是，容量都比较小，所存储的数据是按一定顺序串行写入和读出的，故对每个存储单元的访问与它在存储器中的位置有关。目前常用的器件多为 EEPROM 或快闪存储器，如 93LCXX 或 24LCXX 等系列。

4. 多端口存储器 MPRAM

多端口存储器是为适应更复杂的信息处理需要而设计的一种在多处理机应用系统中使用的存储器。其主要特点是：有多套独立的地址机构（即多个端口），共享存储单元的数据。多端口 RAM 一般可分为双端口 SRAM、VRAM、FIFO、MPRAM 等几类。

其中 VRAM（Video DRAM）视频 RAM 是专门为了图形应用优化的双端口存储器（可同时与 RAM、DAC 以及 CPU 进行数据交换），能有效地防止在访问其他类型的内存时发生的冲突；FIFO 存储器（先进先出存储器）是一种具有存储功能的高速器件，可在高速数字系统中用作数据缓存。FIFO 通常利用双口 RAM 和读写地址产生模块来实现其功能；MPRAM 即高速多端口存储器，主要用于高速数据采集。

5. FPGA 中的嵌入式存储器

在 5.6 节中曾提到过 FPGA 中含有可编辑的嵌入式存储器（Embedded Memory），即嵌入式阵列块 EAB（Embedded Array Block），或嵌入式系统块 ESB（Embedded System Block）。在 Cyclone 和 Cyclone Ⅲ FPGA 中分别称 M4K 和 M9K 模块，其实大同小异，都属于嵌在 FPGA 中的可配置编辑的 SRAM 型存储单元，它们构成了庞大的阵列块。这些嵌入式阵列块可用来实现不同类型的存储器功能，例如每个 EAB 可以提供 2048 位。设计者可以利用 EDA 工具构造一些规模不是太大的，但能高速运行，有不同数据宽度的 SRAM、ROM、FIFO、双口 RAM、移位寄存器等功能块。这些嵌入在 FPGA 中的模块可以作为普通存储器，也可以用于实现复杂的函数的查找表，如正弦、余弦、乘法运算表等。此外，FPGA 中的 EAB 还能用来实现速度远高于普通门电路构成的逻辑模块，如计数器、地址译码器、状态机、乘法器等。通常每个 EAB 可以贡献 100～600 个等效门。

在 Cyclone FPGA 器件中所含的嵌入式存储器，由大量的 M4K 的存储器模块构成，每个 M4K 存储器块具有很强的伸缩性，它本质上就是 EAB。EAB 可以单独使用，也可以组合起来使用。用 EAB 构建不同类型和不同大小的存储器非常方便，这可以从图 9-18 的示意图看出。即根据实际需要，通过软件编辑和参数选择，方便地组织 FPGA 内嵌入的 SRAM 单元，构成不同深度和位宽的 RAM 或 ROM。所谓深度是存放二进制数据的数量，例如所编辑的 RAM 大小是 2048×32 位，即此 RAM 的数据端口的二进制位宽是 32 位，共含 2048 个存储单元，即深度是 2048，每个存储单元有 32 个二进制数据位，对应的地址线宽度是 11 位，但这要 32 个 EAB 组织而成。

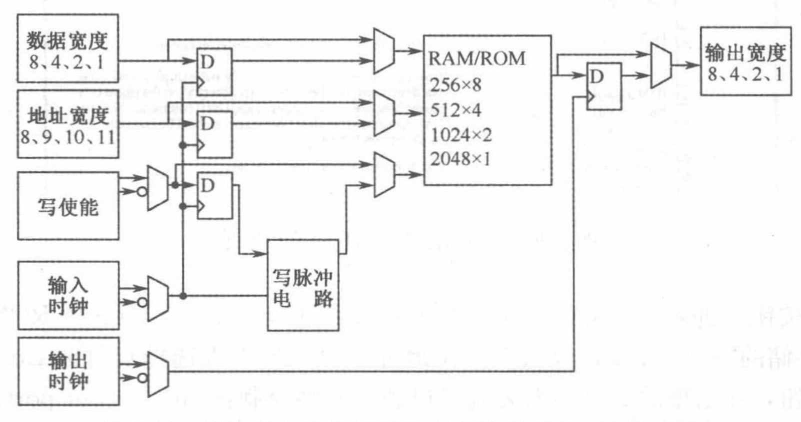


图 9-18 利用 EAB 可配置不同结构容量的 RAM 和 ROM

Altera 公司已经推出的 CycloneⅢ、CycloneⅣ和 CycloneⅤ等系列的 FPGA 中所含的存储器模块的规模更大，速度更高，适用面更宽，使用也更方便。

9.4 存储器应用电路设计

为了便于验证和方便实验，也为了使读者了解基于FPGA的数字系统设计环境中，存储器的使用方法，以下给出两则LPM存储器使用实例。

9.4.1 利用LPM_ROM设计查表式乘法器

高速的硬件乘法器有多种设计方法，可以用逻辑门来实现，也可以用加法器通过移位相加的方式实现，但相比之下，由ROM构成的乘法表查表方式的乘法器的运算速度最快，也比较节省硬件资源。这是因为，已经预先把所有可能的结果都计算出来，并都存放在ROM中了，只待查阅。

这里介绍使用FPGA中的嵌入式存储器构成一个4×4位乘法器的方法。设计流程可以参考第6章，调用LPM模块可以参考第8章。在建立工程和打开原理图编辑窗后，可以进入LPM模块调用窗（图8-31）。然后进入如图9-19所示的对话框，在左栏选择存储器Memory Compiler中的单口ROM：1-PORT，在右栏选择目标芯片系列Cyclone Ⅲ，并选择Verilog HDL，在路径部分键入当前设置元件文件名：MULT4。

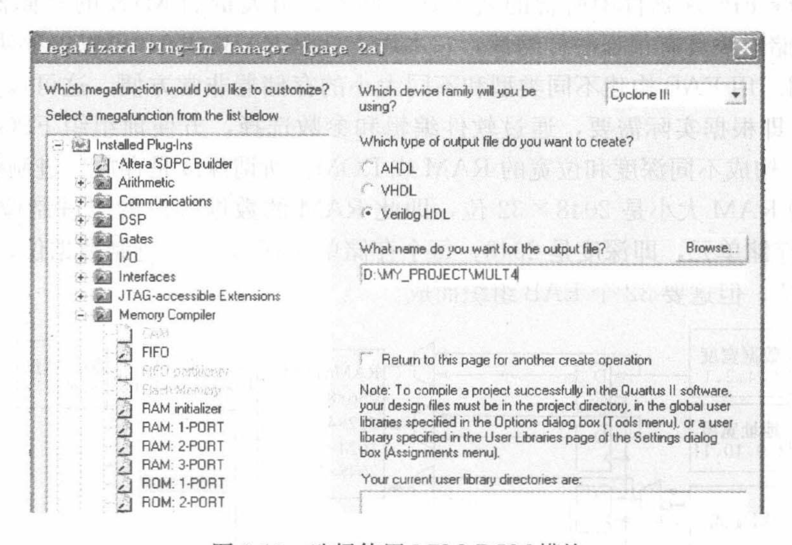


图9-19 选择使用LPM_ROM模块

按Next按钮，进入下一窗口。在接下来的对话框（图9-20）中选择ROM的输出数据为8位；存储的字数是256，表明有8位地址线。在最下方选择双时钟Dual clock。然后按Next按钮，在出现的窗口中消去输出口锁存时钟控制：'q' output port。即地址输入仅由时钟inclock的上升沿锁入（为了能在线监测其内部的数据），数据位宽也为8位。

最后为ROM配置乘法表数据文件。进入图9-21所示的对话框中，在File name栏选择ROM的配置文件rom_data.mif。此LPM_ROM中作为乘法表的数据文件已预先编

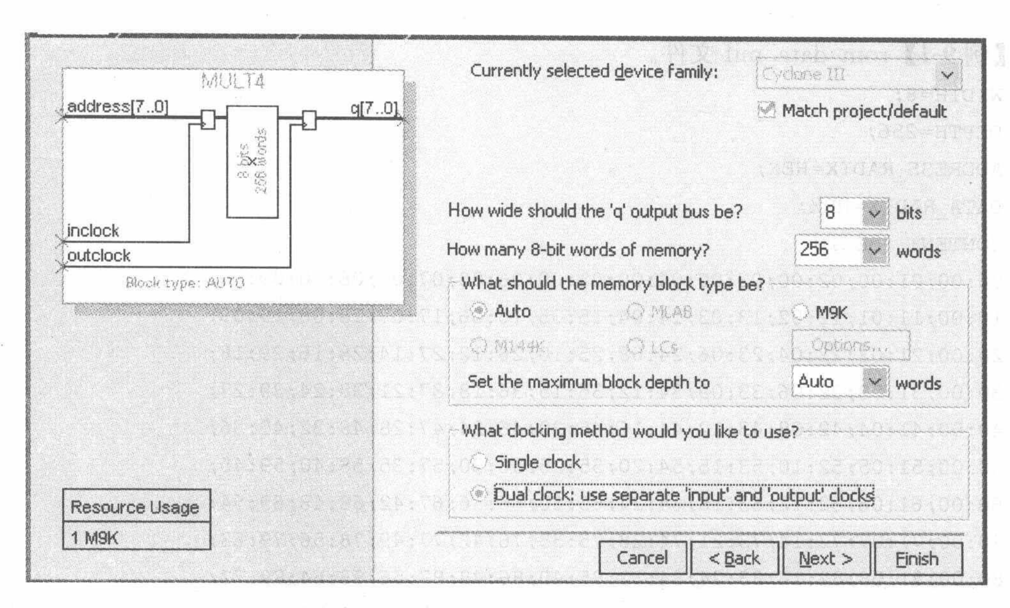


图 9-20 对 LPM_ROM 模块设置必要的参数

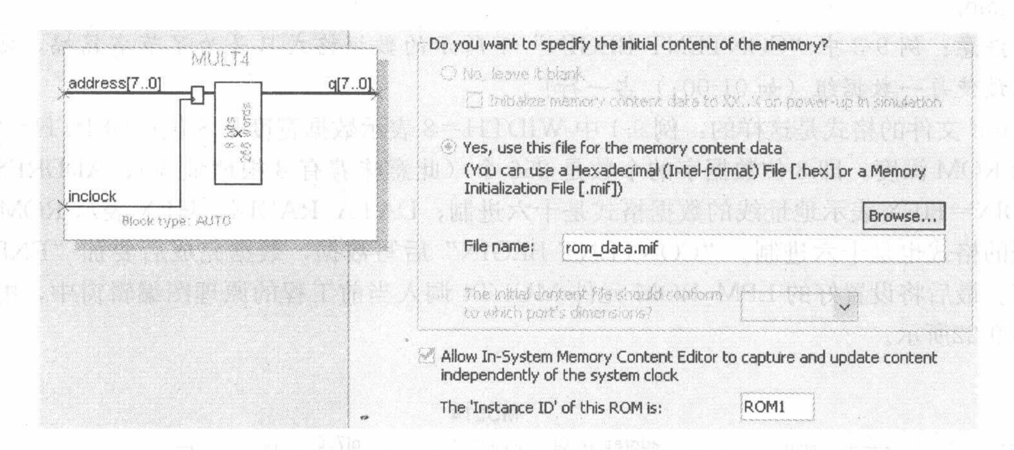


图 9-21 为 LPM_ROM 选择初始化配置文件 rom_data. mif

辑好了，例如取文件名为 rom_data. mif，文件如例 9-1 所示。

在图 9-21 下面若选中 Allow In-System Memory…复选框，并在 The Instance ID of this ROM is 文本框中输入 ROM1，作为此 ROM 的 ID 名称。通过这个设置，可以允许 QuartusⅡ通过 JTAG 口对下载于 FPGA 中的此 ROM 进行“在系统”测试和读写。

QuartusⅡ中 LPM_ROM 宏模块能接受的初始化数据文件的格式有两种：Memory Initialization File（.mif）格式和 Hexadecimal（Intel-Format）File（.hex）格式。实际应用中只要使用其中一种格式的文件即可。

数据文件 rom_data.mif 中的地址/数据表达方式是（例 9-3）：冒号左边写 ROM 地址值，冒号右边写对应此地址放置的十六进制数据。如 47:28，表示 47 为地址，28 为该地址中的数据，这样，地址高 4 位和低 4 位可以分别看成是乘数和被乘数，输出的数据可以看成是它们的乘积，即 4×7=28。

【例 9-1】 rom_data. mif 文件。

```
WIDTH=8;
DEPTH=256;
ADDRESS_RADIX=HEX;
DATA_RADIX=HEX;
CONTENT  BEGIN
00:00;01:00;02:00;03:00;04:00;05:00;06:00;07:00;08:00;09:00;
10:00;11:01;12:02;13:03;14:04;15:05;16:06;17:07;18:08;19:09;
20:00;21:02;22:04;23:06;24:08;25:10;26:12;27:14;28:16;29:18;
30:00;31:03;32:06;33:09;34:12;35:15;36:18;37:21;38:24;39:27;
40:00;41:04;42:08;43:12;44:16;45:20;46:24;47:28;48:32;49:36;
50:00;51:05;52:10;53:15;54:20;55:25;56:30;57:35;58:40;59:45;
60:00;61:06;62:12;63:18;64:24;65:30;66:36;67:42;68:48;69:54;
70:00;71:07;72:14;73:21;74:28;75:35;76:42;77:49;78:56;79:63;
80:00;81:08;82:16;83:24;84:32;85:40;86:48;87:56;88:64;89:72;
90:00;91:09;92:18;93:27;94:36;95:45;96:54;97:63;98:72;99:81;
END;
```

注意： 例 9-3 中“CONTENT BEGIN”下所示的数据格式只是为了节省篇幅，实用中应该使每一数据组（如 01:00;）占一行！

mif 文件的格式是这样的：例 9-1 中 WIDTH＝8 表示数据宽度是 8 位；DEPTH＝256 表示 ROM 深度，即 8 位数据字的个数是 256 个（此意味着有 8 根地址线）；ADDRESS_RADIX＝HEX 表示地址线的数据格式是十六进制；DATA_RADIX＝HEX 表示 ROM 中数据的格式也是十六进制。“CONTENT BEGIN”后写数据，数据完成后要加“END;”结束。最后将设置好的 LPM_ROM 元件 MULT4 调入当前工程的原理图编辑窗中，电路如图 9-22所示。

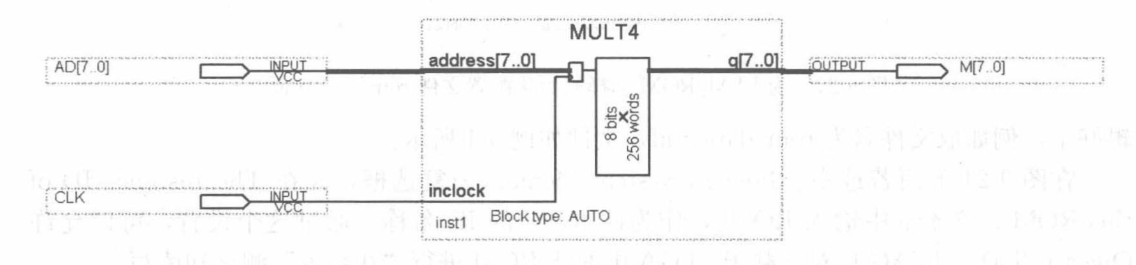


图 9-22　ROM 乘法器测试电路

目标器件可选 EP3C10E144C8 或其他系列 FPGA。首先全程编译，然后进行时序仿真。仿真波形如图 9-23 所示。从图中可以看出，在 CLK 的上升沿，将 4 位乘数和被乘数的数据当成地址数据锁入 ROM 中，而“运算”的结果，以 ROM 数据端输出数据的方式输出。

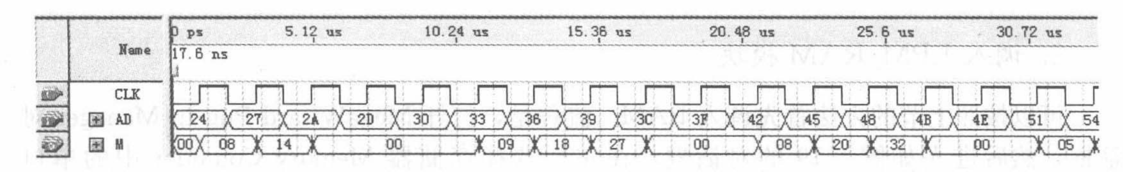


图 9-23 ROM 乘法器时序仿真波形

例如在 CLK 的第二个上升沿，AD=27，对应输出 14。尽管它们是十六进制，但完全可以看成是十进制数，即 2×7=14；同理有：3×3=09，3×6=18，3×9=27，4×8=32 等。而当输入有大于 9 的值出现时，如 2A、2D、3C、3F、4E 等，输出都为 0。

最后可以在锁定引脚后下载到 FPGA 中进行硬件测试。此例只是作为一个用存储器实现查表式逻辑功能的说明，其实在逻辑设计乃至普通电子设计中，这种基于 ROM 的查表式功能模块有很广阔的适用领域。

9.4.2 多通道数字信号采集电路设计

逻辑分析仪是一个多通道数字信号数据采样、显示与分析的电子设备。逻辑分析仪可以将数字系统中的脉冲信号，逻辑控制信号，总线数据，甚至毛刺脉冲都能同步高速地采集进该设备中的高速 RAM 中暂存，以备显示和分析。因此逻辑分析仪在数字电路、数字系统、计算机的设计开发和科研中提供了必不可少的帮助。

本示例只是利用 RAM 和一些辅助器件设计一个多通道数字信号采集电路模块。但如果进一步配置好必要的控制电路和通信接口，确能构成一台实用的设备。

1. 基本电路结构

图 9-24 是一个 8 通道逻辑数据采集电路，主要由 3 个功能模块构成：一个随机数据存储器 LPM_RAM、一个 10 位计数器 LPM_COUNTER，以及一个锁存器 74244。

RAM0 是一个 8 位存储器，存储 1024 个字节，有 10 根地址线 address[9..0]，它的 data[7..0] 和 q[7..0] 分别是 8 位数据输入和输出总线口；wren 是写入允许控制，高电平有效；inclock 是数据输入锁存时钟；inclocken 是此时钟的使能控制线，高电平有效。

为了构建图 9-24 的电路，首先利用 QuartusⅡ建立了一个原理图工程。

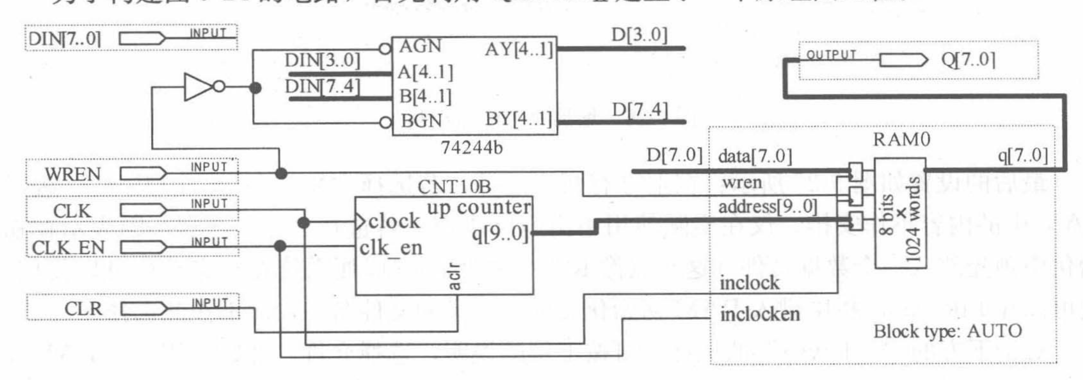


图 9-24 逻辑数据采样电路顶层设计

2. 调入 LPM_RAM 模块

可以按照上节的方法首先调入 RAM。流程是，打开 MegaWizard Pug-In Manager 对话框，然后进入如图 9-19 的对话框，在左栏选择存储器 Memory Compiler 中的单口 RAM：1-PORT，在右栏选择目标芯片系列 Cyclone Ⅲ，并选择 Verilog HDL，在路径部分键入当前设置元件文件名，如 RAM0。然后进入图 9-25 所示窗口。

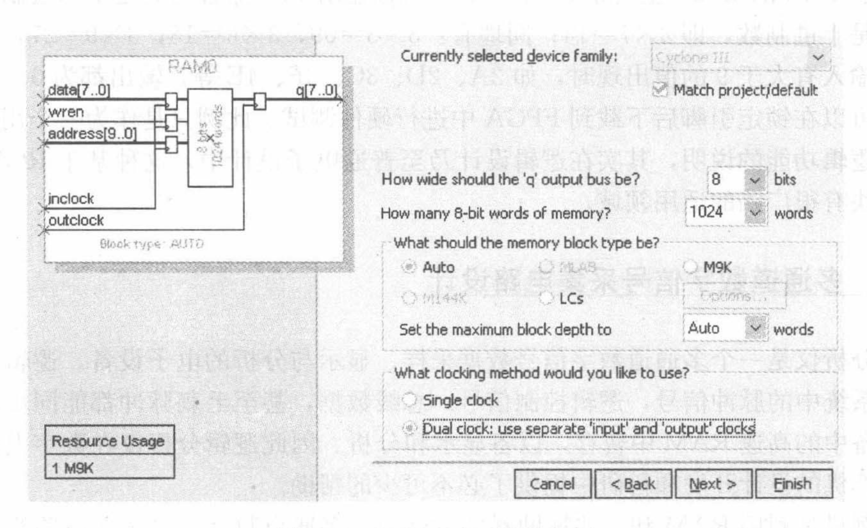


图 9-25 LPM_RAM 参数设置

在图 9-25 窗口中选择数据输出总线是 8 bits，存储字节数是 1024，选择时钟方式是分开的双时钟形式，即 inclock 和 outclock。在下一个窗口（图 9-26）中删除数据输出控制时钟 outclock（理由同上例），再增加一个时钟使能控制端 inclocken。

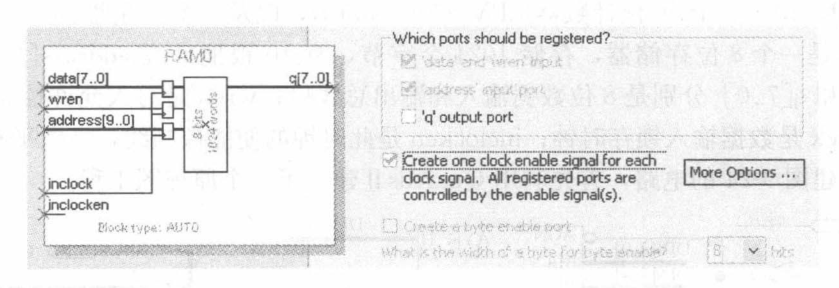


图 9-26 增加时钟使能控制

最后的设置如图 9-27 所示：在上方有两个选择，若选择“No，leave it blank”，表示 RAM 中的内容不作安排，仅在实际使用中由电路决定；若选择“yes，…”，则表示在初始化中预先放入一个数据文件（这有点像 ROM 的功能），以便系统在一启动后可以使用。这可以在 File name 栏中键入 RAM 初始化文件的路径和文件名。在此不作此选择。

对于下方的“Allow…”的选择，可按上例的说明，选择允许，并键入 ID 名:RAM1。最后将此 RAM0 元件调入当前工程的原理图编辑窗。

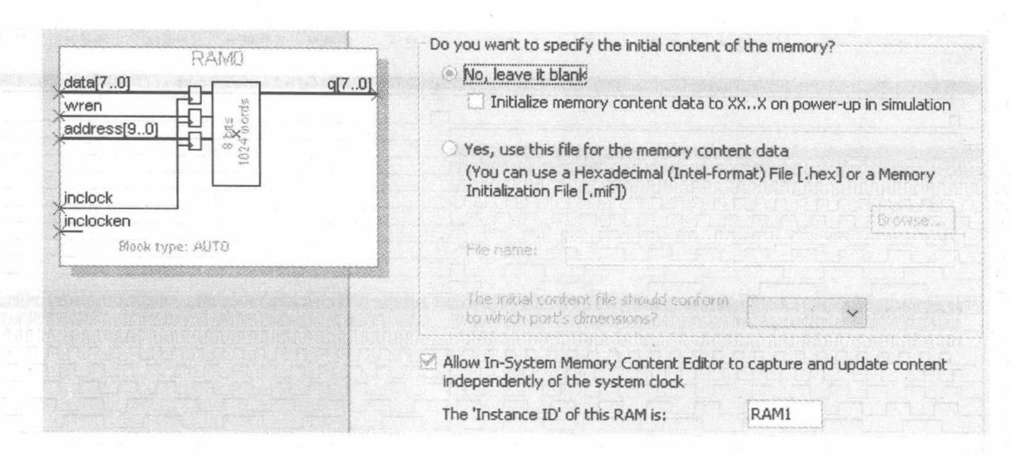


图 9-27 允许在系统存储器内容编辑器能对此 RAM 编辑和测试

3. 调入计数器模块 LPM_COUNTER

LPM_COUNTER 计数器模块的调用方法可以参考 8.5 节。最后一个调入的元件是 74244b。完成后，按照图 9-24 连接好电路图。

4. 系统功能分析

编译完成后准备时序仿真，测试此电路系统的功能。在此之前可以初步分析一下此电路的基本功能，便于在 VWF 仿真激励文件中正确设置激励信号（波形）。

图 9-24 中，10 位计数器 CNT10B 主要用作此存储器的地址信号发生器，由外部 CLK 同步控制计数器和存储器的计数速度，即采样速度。74244b 主要起到一个隔离的作用。当 wren 为高电平时，外部数据通过 74244 进入 RAM；当 wren 为低电平时，RAM 禁止输入，74244b 的输出口呈高阻态。如果要读出 RAM 中已存入的数据，在 wren 为低电平的条件下，必须使 CLR 产生一个高电平脉冲，对计数器清 0，以便对 RAM 中的数据从地址的最低端读起。然后启动 CLK 即可读出所存数据。

5. 系统时序仿真

对图 9-24 电路的时序仿真报告波形图如图 9-28 所示。注意对激励信号，即输入信号 CLK、CLK_EN、CLR、WREN 和输入总线数据 DIN[7..0] 的激励信号波形的设置及时序安排。时序仿真中必须注意，正确设置激励信号是成功完成系统时序分析的关键。如果在一开始，对电路的基本功能毫不知情，只是随意设置一些仿真波形数据（激励信号）于输入端，则几乎不可能获得正确的电路分析结果。

图 9-28 的激励信号设置情况是这样的：仿真时间轴设为 50us（微秒）；CLK 的频率可以设得高一点，以便向 RAM 中输入更多的数据，周期设为 80ns；CLK_EN 全程（及对 RAM 的读和写）都设为高电平，都允许工作；CLR 在数据写入和读出前都发一个脉冲，以便对地址发生计数器清 0；WREN 在 RAM 写入段设为高电平（写允许），在 RAM 读出段设为低电平（写禁止）。对于 RAM 数据输入口的 8 根输入线，分别设置好不

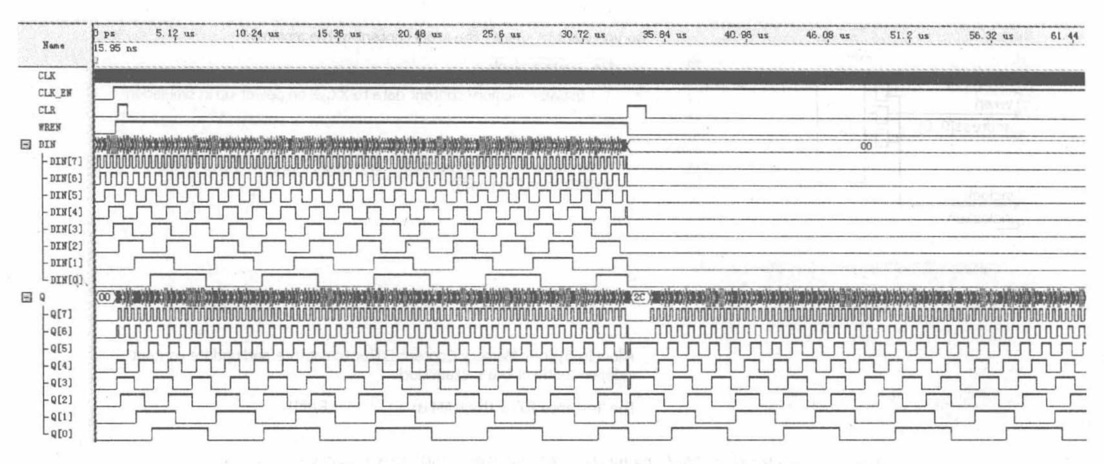


图 9-28　逻辑数据采样电路时序仿真波形

同频率的待采样信号，且最好这些信号仅在数据写入段存在（图 9-28）。

由图 9-28 可见，在 RAM 数据读出时间段，能正确地将写入的数据完整地按地址输出。这表明，图 9-24 的电路确能成为一个 8 通道的数字信号采集系统。

接下去的工作就是对此项设计进行硬件测试。首先根据实验系统的基本情况进行引脚锁定，编译，然后下载。被测的 8 路逻辑信号可以来自实验系统上的时钟信号源。

展示已采样进入 LPM_RAM 中的信号有多种方式。如利用 QuartusⅡ的在系统存储器内容编辑器 In-System Memory Content Editor，或使用 QuartusⅡ的 SignalTapⅡ。

习　题

9-1　ROM 和 RAM 各适用于什么场合？它们由哪几个部分组成？各有什么作用？

9-2　试比较 ROM、PROM 和 EPROM 及 E^2PROM 有哪些异同。

9-3　试画出用 1024×4 位的 RAM 扩展成 4096×4 位的 RAM 接线示意图。

9-4　用 128×8 位的 ROM 实现不同码制间的转换。要求用从全 0 地址开始的前 16 个地址单元实现 8421BCD 码到余 3 码的转换；接下来的 16 个地址单元实现余 3 码到 8421BCD 码的转换：

(1) 列出 ROM 的地址与内容对应关系的真值表；

(2) 确定输入变量和输出变量与 ROM 地址线和数据线的对应关系；

(3) 简要说明将 8421BCD 码的 0101 转换成余 3 码和将余 3 码转换成 8421BCD 码的过程。

9-5　试用 256×8 位的 RAM 若干片构成一个 1024×8 位的 RAM，求需要多少片？画出连接图。

9-6　试用 2048×8 位的 RAM 6116 集成芯片若干片，构成一个 8192×16 位的 RAM，求需要多少片？画出连接图。若扩展成 1024×8 位 RAM 需要多少块 256×4 位 RAM？画出连接图。

9-7　ROM 256×8 位的存储器有多少根地址线、字线和位线？试用它实现一个 4 位加

法器。

9-8 按存取方式的不同比较 SRAM、DRAM、ROM 器件的异同点。

9-9 试用 SRAM 6264（8K×8 位）静态随机存储器构建一个能存储 64K 字长为 8 位的存储系统。

9-10 指出下列存储系统各具有多少个存储单元，至少需要几根地址线和数据线，若采用 32K×1 位的存储芯片，各需要多少芯片？

(1) 64K×1　(2) 256K×4　(3) 1M×1　(4) 128K×8

9-11 试用 PROM 实现各逻辑函数，并画出相应电路图。已知逻辑函数 F_1～F_4 分别为

$F_1(A, B, C, D)=\overline{ABD}+BD+A\overline{B}C$

$F_2(A, B, C, D)=A\,\overline{BC}+\overline{A}\,B\overline{C}+\overline{AB}C+A\,C\overline{D}+\overline{C}D$

$F_3(A, B, C, D)=\overline{A}\,C\overline{D}+A\overline{C}D+\overline{A}BD+\overline{BD}$

$F_4(A, B, C, D)=A\,\overline{BC}+\overline{A}BC+B\overline{C}D+\overline{B}CD$



实验

9-1 查表式硬件运算器设计

(1) 按照 9.4.1 节的流程，设计一个 4×4 位查表式乘法器。包括创建工程，调用 LPM_ROM 模块 MULT4，在原理图编辑窗中绘制电路图，全程编译，对设计进行时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译，编程下载于 FPGA 中，进行硬件测试。完成实验报告。

(2) 在以上实验的基础上，增加一些电路（必要时可以复用模块 MULT4），完成一个 8×8 位查表式乘法器的设计。根据以上实验的要求，完成完整的实验流程。

(3) 利用 ROM 的查表完成算法的原理，将 8 位二进制数转换为 3 位十进制数。要求调用的 LPM ROM 有 8 根地址线，作二进制数输入口；输出数据线有 10 根，两个低 4 位分别接两个 7 段译码器输出显示两个低位十进制数，ROM 的另两个高位接第三个 7 段译码器驱动数码管显示最高位十进制数。

用最快捷的方法获取 mif 文件。对设计进行时序仿真，根据仿真波形说明此电路的功能，进行引脚锁定编译，编程下载于 FPGA 中，进行硬件测试。完成实验报告。

9-2 简易逻辑分析仪设计

(1) 按照 9.4.2 节的流程，设计一个 8 通道，深度为 2048 的简易逻辑分析仪。包括创建工程，调用 LPM_RAM 等模块，在原理图编辑窗中绘制电路图，全程编译，对设计进行时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译、编程下载于 FPGA 中，进行硬件测试。完成实验报告。

8 个通道被采样的数字信号可以用实验系统上的时钟信号源代替。通过实验系统上的控制键，可以将采入 FPGA 内 LPM_RAM 中的数据显示到数码管、发光管或示波器上。

当然也可以利用 Quartus Ⅱ 的 In-System Memory Content Editor，直接显示在屏幕上；或使用 Quartus Ⅱ 的 SignalTap Ⅱ 来验证，它们的用法可以查阅本书参考文献[1]。

（2）为以上设计增加一些控制，使之更加完善。例如增加一些逻辑，控制图 9-24 的 WREN、CLK_EN、CLR 等，使此系统含有不同的采样触发模式和触发信号来源。如能手动方式或自动方式触发采样，即每接收到一个脉冲信号（可以来自键控或外部启动信号），即进行一次采样，或多次采样，或连续采样（一次采样深度必须是 2048 位）；而触发脉冲的触发方式可以预先设定，如高电平触发、低电平触发、上升沿触发、下降沿触发等。

第10章 D/A与A/D转换器及其应用

本章部分内容将涉及诸如采样定理、运算放大器、模拟比较器等概念或电路元件，对于这些内容尚无相关基础知识的读者（例如在第一学期就使用本教材学习的读者）或许会感到陌生，但可以肯定，这并不会影响对A/D和D/A的整体理解和使用方法的掌握。对于陌生的内容可以暂时放过，以待在本章给出的实验中或后续课程中带着问题深入学习或通过查阅有关资料来深入了解。本章学习的重点是了解A/D和D/A的工作原理、结构特点、电气参数和基本使用方法，学习的任务是能利用前面已掌握的数字技术正确使用A/D和D/A，进而能通过实践，自主设计出一些要用到A/D和D/A的有创新意义的实用数字系统。

10.1 概　述

一个典型的计算机检测与控制系统的框图如图10-1所示。在图中，生产过程中的各种物理量，通过传感器变成模拟电信号，通过A/D转换器变成数字信号；经过计算机处理后，通过D/A转换器将数字信号变成模拟信号，对生产过程进行控制。

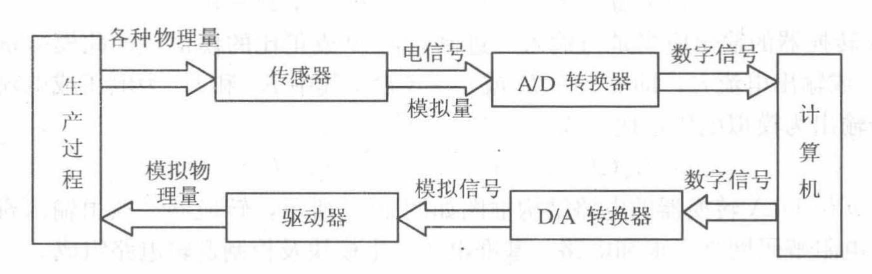


图10-1　计算机检测与控制系统示意图

显然，电子技术的发展使得现代数字系统与模拟电子系统变成了两个独立的世界，它们可以被独立研究，单独设计。它们之间的最重要的联系只剩下模/数转换器和数/模转换器了。在绝大多数情况下，独立数字系统是无法构成实用电子产品的，它们必须通过A/D和D/A来和真实的模拟世界进行联系。这就有必要熟悉并掌握数字系统如何通过A/D和D/A与模拟电路或其他应用电路接口或连接的方式。

数字系统在人们的日常生活和生产当中的应用已相当普及，大量的数字测控产品以及各类数字仪表在许多领域中的广泛应用导致数字技术处理模拟信号的情况更加普遍，从而在技术上模拟信号与数字信号的接口问题就变得更加突出和重要。

为了能够使数字系统有效地处理模拟信号，必须把模拟信号转换成数字信号才能进行数字处理，这就需要用到模/数转换器。另一方面，也需要把处理后的数字信号再转换成模拟信号，对生产过程进行控制，这就需要用到数/模转换器。

10.2 D/A 转换器

能把模拟量转换为数字量的电路称为模/数转换器（Analog to Digital Converter），简称 A/D 转换器或 ADC；而能把数字量转换为模拟量的电路则称为数/模转换器（Digital to Analog Converter），简称 D/A 转换器或 DAC。现代电子系统中，正是 ADC 和 DAC 构成了沟通模拟电路和数字电路主要的桥梁。本节首先介绍 D/A 转换器的工作原理和电路结构，说明 D/A 转换器的模拟输出与数字输入之间的关系；然后介绍 D/A 转换器的主要技术参数，并给出两则经典集成 D/A 转换器的基本原理及其使用方法；本章最后安排了对 DAC 的实验项目。

10.2.1 D/A 转换原理

对于输入的二进制数代码，DAC 对它的每一位按其权的大小转换成相应的模拟量，然后将代表各位的模拟量相加，所得的总模拟量就与数字量成正比，这样便实现了从数字量到模拟量的转换。

设 D/A 转换器的输入是一个 n 位二进制数 D（$d_{n-1}d_{n-2}\cdots d_1d_0$），将每一位按权展开后可得

$$D = d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + \cdots + d_1 2^1 + d_0 2^0 \tag{10-1}$$

D/A 转换器的输出应当是与输入二进制数值 D 成正比的模拟量，此模拟量可以是输出电压 u_o 或输出电流 i_o，即 $u_o = K_u D$ 或 $i_o = K_i D$，其中 K_u 和 K_i 为电压或电流转换比例系数。若输出为模拟电压，代入式（10-1）后得

$$u_o = K_u(d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + \cdots + d_1 2^1 + d_0 2^0) \tag{10-2}$$

通常 n 位 D/A 转换器的电路结构框图如图 10-2 所示，转换器一般由输入寄存器、电子开关、电阻解码网络、求和电路、基准电压产生模块及控制逻辑电路组成。



图 10-2 D/A 转换器的电路结构框图

输入的数字信号可以以并行或串行方式输入，输入的数据保存在输入寄存器里。此寄存器以并行方式输出，其数据用来控制每一位电子开关，使电阻解码网络将每一位数码转换成相应大小的模拟量，并送给求和电路，求和电路由一个运算放大器担任。求和电路将各位数码所代表的模拟量相加，从而得到数字量 D 所对应的模拟量 A。

目前常见的 D/A 转换器解码网络有多种类型，其中包括二进制权电阻网络型、倒 T

型电阻网络型、权电流型、权电容网络型、流水线结构型，以及开关树型 D/A 转换器等多种类型。本章仅介绍权电阻网络和倒 T 型电阻网络两类 D/A 转换器。

10.2.2 二进制权电阻网络型 D/A 转换器

这是一种比较早期的 DAC，结构简单，工作原理典型，但实现工艺要求较高。其原理示意型的 4 位二进制权电阻网络 D/A 转换器结构如图 10-3 所示。整个电路由基准电压 V_{REF}、电子开关 S_0～S_3、权电阻网络和运算放大器 A 组成。电路的输入数据是 D（$d_3d_2d_1d_0$）；输出是模拟电压 V_O。

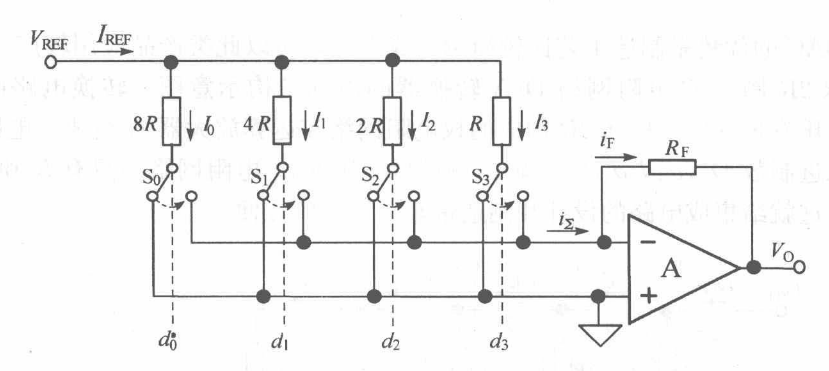


图 10-3 二进制权电阻网络 D/A 转换器

在此，完全可以把运算放大器 A 看成是一个模拟电压加法器，当电子开关 S_0～S_3 中某一开关打开时，则对应的电压量就通过运算放大器 A，被加入到输出电压 V_O 了，其中的 R_F 是输出电压大小的比例控制电阻。

根据电路中利用运算放大器同相输入端接地，则反相输入端为“虚地”的原理，数字信号由输入端 d_3、d_2、d_1、d_0 并行输入，分别控制电子开关 S_3、S_2、S_1、S_0；各权电阻的公共端接基准电压 V_{REF}。当数字量 d_i 为 1 时，开关 S_i 接集成运放反向输入端，有支路电流 I_i 流向求和放大电路；当数字量 d_i 为 0 时，开关接地，支路电流 I_i 为 0。显然不论电子开关接到运算放大器的反相输入端（虚地）还是接到地，也就是说不论输入数字信号是 1 还是 0，各支路的电流是不变的，因此有

$$I_0=\frac{V_{REF}}{8R}\quad I_1=\frac{V_{REF}}{4R}\quad I_2=\frac{V_{REF}}{2R}\quad I_3=\frac{V_{REF}}{R}$$

$$i_{\Sigma}=I_0+I_1+I_2+I_3=\frac{V_{REF}}{8R}D_0+\frac{V_{REF}}{4R}D_1+\frac{V_{REF}}{2R}D_2+\frac{V_{REF}}{R}D_3$$

$$=\frac{V_{REF}}{2^3R}(2^3\cdot D_3+2^2\cdot D_2+2^1\cdot D_1+2^0\cdot D_0)\tag{10-3}$$

设 $R_F=R/2$，由式（10-3）可得

$$V_O=-R_Fi_F=-\frac{R}{2}\cdot i_{\Sigma}=-\frac{V_{REF}}{2^4}(2^3\cdot D_3+2^2\cdot D_2+2^1\cdot D_1+2^0\cdot D_0)\tag{10-4}$$

由式（10-4）可得输出的模拟电压正比于输入的二进制数，故实现了数字量与模拟量

的转换。且当输入的数字量为 n 位时，输出电压的最大变化范围是

$$0 \sim -\frac{2^n-1}{2^n}V_{REF}$$

此类DAC的缺点是所含电阻的种类多、各电阻阻值都不同。尤其当输入信号的位数较多时，问题便更突出；因为为了实现数字量到模拟量的精确转换，就要求每个电阻都有很高的精度，这势必增加其生产成本和集成制造工艺难度。为了克服这一缺点，通常采用以下介绍的倒T型电阻网络D/A转换器。

10.2.3 倒T型电阻网络D/A转换器

这类DAC的优势是制造工艺比较简单，成本低，所以此类产品应用较广。图10-4给出了4位R-2R倒T型电阻网络D/A转换器的内部结构示意图。转换电路由基准电压 V_{REF}、电子开关 $S_0 \sim S_3$、R 和 $2R$ 构成的权电阻网络和运算放大器A组成。电路的输入数据是4位二进制数 D（$d_3d_2d_1d_0$），输出是模拟电压 V_O。电阻网络中只有 R 和 $2R$ 两种阻值的电阻，这就给集成电路的设计和制造带来了很大的方便。

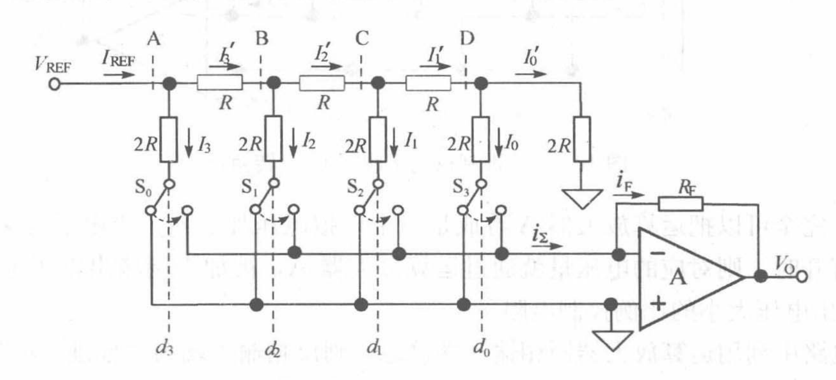


图10-4 R-2R倒T型电阻网络D/A转换器

仍然根据以上提到的运算放大器的虚地原理，R-2R倒T型电阻网络的特点是：

(1) 分别从虚线A、B、C、D处向右看的二端网络等效电阻都是 R。

(2) 不论模拟开关接到运算放大器的反相输入端（虚地）还是接到地，即，不论输入数字信号是1还是0，各支路的电流都不变。

当数字量为1时，开关接集成运放反向输入端，有支路电流 I_i 流向求和放大电路；当数字量为0时，开关接地，支路电流 I_i 为0。于是由图10-4可知，来自参考电压端输入的电流为 $I_{REF}=\frac{V_{REF}}{R}$，其他各段的电流为

$$I_3=\frac{1}{2}I_{REF}=\frac{V_{REF}}{2R}, I_2=\frac{1}{4}I_{REF}=\frac{V_{REF}}{4R}, I_1=\frac{1}{8}I_{REF}=\frac{V_{REF}}{8R}, I_0=\frac{1}{16}I_{REF}=\frac{V_{REF}}{16R}$$

求和运算放大器的输出电压应该是

$$V_O=-R_Fi_F=-R_Fi=-\frac{V_{REF}R_F}{2^4R}(2^3\cdot d_3+2^2\cdot d_2+2^1\cdot d_1+2^0\cdot d_0) \quad (10\text{-}5)$$

当 $R_F=R$ 时，

$$V_O=-\frac{V_{REF}}{2^4}(2^3\cdot d_3+2^2\cdot d_2+2^1\cdot d_1+2^0\cdot d_0) \qquad (10\text{-}6)$$

此式表明，输出的模拟电压正比于输入的二进制数，故实现了数字量与模拟量的转换。倒T型电阻网络D/A转换器的突出优点是：无论输入信号如何变化，流过基准电源及各支路的电流始终不变，因此不需要电流建立时间，这有利于提高转换速度。电阻网络中的电阻只有 R 和 $2R$ 两种取值，便于集成制造。倒T型电阻网络D/A转换器是目前应用较多的转换电路。

【例 10-1】 4位R-2R倒T型电阻网络DAC如图10-4所示，设基准电压 $V_{REF}=-8V$，$R_F=R$，试求其最大输出电压值。

解：将 $d_3d_2d_1d_0=1111$ 代入式（10-6），得出最大输出电压值为7.5V：

$$V_O=-\frac{V_{REF}}{2^4}(2^3\cdot d_3+2^2\cdot d_2+2^1\cdot d_1+2^0\cdot d_0)$$

$$=-\frac{-8V}{2^4}(2^3\cdot 1+2^2\cdot 1+2^1\cdot 1+2^0\cdot 1)=7.5V$$

10.2.4 D/A转换器的主要技术参数

D/A转换器有多个技术指标，但主要技术指标是：分辨率、转换精度和转换速度。

1. 分辨率

DAC的分辨率用输入二进制数的有效位数表示。在分辨率为 n 位的DAC中，输出电压能区分 2^n 个不同的输入二进制代码状态，能给出 2^n 个不同等级的输出模拟电压。分辨率也可以用D/A转换器的最小输出电压 V_{LSB}（输入数字只有最低位为1）与最大输出电压 V_{FSR}（输入数字全为1）的比值来表示。n 位DAC的分辨率为

$$\text{分辨率}=1/(2^n-1) \qquad (10\text{-}7)$$

例如，10位D/A转换器的分辨率为

$$\frac{1}{2^n-1}=\frac{1}{2^{10}-1}=\frac{1}{1023}\approx 0.001$$

显然位数 n 越大，其输出模拟电压的取值个数就越多（2^n 个）或取值间隔（2^n-1 个）越多，于是D/A输出模拟电压的变化量就越小，就越能反映输出电压的细微变化。

2. 转换精度

D/A转换器的转换精度是指输出模拟电压的实际值与理想值之差，即最大静态转换误差。通常要求D/C转换器的误差小于 $V_{LSB}/2$。

3. 转换速度

转换速度通常用转换时间来表示。转换时间是指从输入数字起，到输出电压或电流到达稳定值时所需要的时间，也称输出建立时间。

4. 非线性误差

通常把D/A转换器输出电压值与理想输出电压值之间偏差的最大值定义为非线性误差。D/C转换器的非线性误差主要由模拟开关以及运算放大器的非线性引起。

5. 温度系数

在输入不变的情况下，输出模拟电压随温度变化而变化的量，称为DAC的温度系数。一般用满刻度的百分数表示温度每升高1度输出电压变化的值。

10.2.5 DAC专用器件及其应用

DAC专用器件有很多种类。以数据输入方式分类，可分为串行输入和并行输入；以输出方式分类，可分为电流输出型和电压输出型；以输出通道分类，有单通道输出和多通道输出类型。这都必须根据实际情况来选用，此外还必须考虑价格、封装、控制端口情况、转换速率、转换精度、工作电平（有TTL电平或3.3V电平等）等因素。

以下将分别简要介绍常用D/A转换器DAC0832、AD7520和TLC5615的基本情况及使用方法。

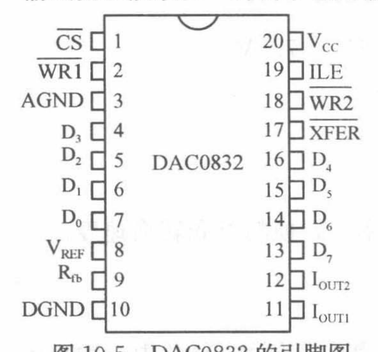


图10-5 DAC0832的引脚图

1. 并行D/A转换器DAC0832的用法

DAC0832是一种非常常用且结构经典的D/A转换器，它是采用CMOS工艺制成的单片电流输出型8位D/A转换器。DAC0832的引脚图如图10-5所示。

DAC0832的引脚功能说明如下：

- $\overline{WR1}$和$\overline{WR2}$分别为输入数据选通信号和数据传送选通信号，都是低电平有效。
- $\overline{XFER}$是数据传送选通信号，低电平有效。
- ILE是输入锁存允许信号，高电平有效。
- $\overline{CS}$是片选信号，低电平有效。
- D_7～D_0是8位输入数据信号。
- V_{REF}是参考电压输入。一般此端外接一个精确、稳定的电压基准源，可在－10～＋10V范围内选择。
- R_{fb}是反馈电阻（内已含一个反馈电阻）接线端。
- I_{OUT1}是DAC输出电流1端，此输出信号一般作为运算放大器的一个差分输入信号。当DAC寄存器中的各位为1时，电流最大；为全0时，电流为0。I_{OUT2}是DAC输出电流2端。它作为运算放大器的另一个差分输入信号（一般接地）。I_{OUT1}和I_{OUT2}满足关系：$I_{OUT1}+I_{OUT2}=$常数。
- V_{CC}是电源输入端（＋5～＋15V）。
- DGND和AGND分别是数字地和模拟地。

DAC0832输出的转换结果是电流。为了将电流转换为电压，还必须经过一个外接的运算

放大器。在DAC芯片内部已设置了一个反馈电阻R_{fb}，只要将R_{fb}脚接到运算放大器的输出端即可。若需增大运算放大器增益，可外加一个反馈电阻与R_{fb}串联。

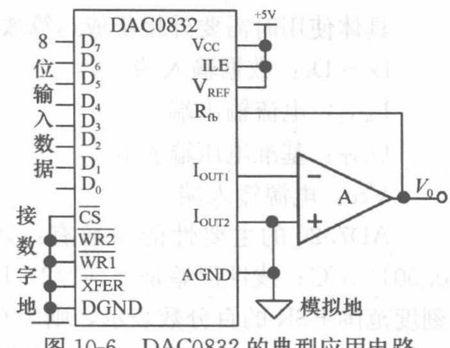


图 10-6　DAC0832 的典型应用电路

图10-6是DAC0832的典型应用电路，图中采用直通方式输出。需要转换的数字信号通过D_0～D_7送入DAC0832，经转换后的输出电流信号I_{OUT1}和I_{OUT2}，接由运放构成的电路，将电流变为电压输出（式中R为R-2R电阻网络中的电阻）：

$$I_{OUT1}=\frac{V_{REF}}{R}\times\frac{(D)_{10}}{256}\qquad I_{OUT2}=\frac{V_{REF}}{R}\times\frac{255-(D)_{10}}{256}\qquad V_O=-(I_{OUT1}\times R_{fb})$$

DAC0832内部有两个寄存器，即输入寄存器和DAC寄存器，在连接上有三种方式：双缓冲、单缓冲和直通方式：

（1）双缓冲方式。采用二级缓冲方式，可在输出的同时，采集下一个数据，提高了转换速度；也可在多个转换器同时工作时，实现多通道D/A的同步转换输出。

（2）单缓冲方式。适合在不要求多片D/A同时输出时。此时只需一次写操作，就开始转换。

（3）直通方式。输出随输入数据的变化而随时转换，图10-6所示即此种方式。

为了减少外部工频信号或其他感应信号的干扰，如图10-6所示，DAC0832电路的模拟地AGND和数字地DGND是分开连接的。这在PCB电路板布线时必须考虑，即在这个电路板上将所有相关器件的模拟地和数字地都必须分别连接，最后再将总的模拟地和总的数字地在一单一点上连在一起。

2. 并行D/A转换器AD7520的用法

AD7520是10位的D/A转换集成芯片，与微处理器完全兼容。该芯片以接口简单、转换控制容易、通用性好、性能价格比高等特点曾得到广泛的应用。

AD7520内部逻辑结构如图10-7所示，AD7520外引脚图如图10-8所示。该芯片内部只含倒T型电阻网络、电流开关和反馈电阻，不含运算放大器。输出端为电流输出。

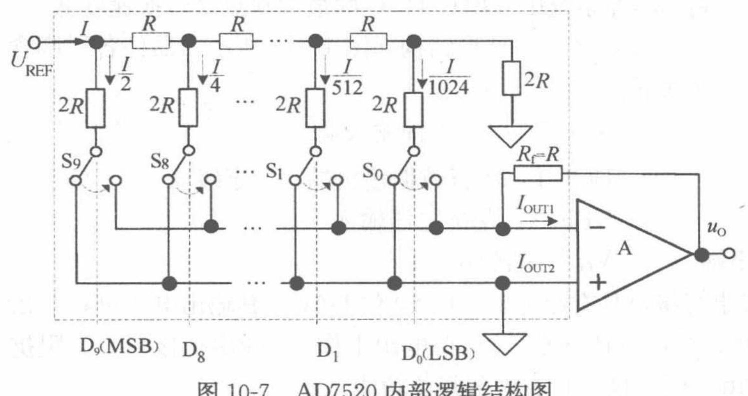


图 10-7　AD7520 内部逻辑结构图

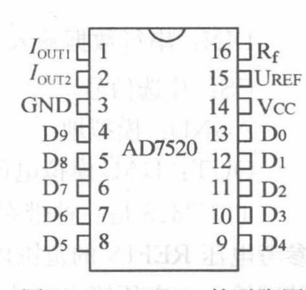


图 10-8　AD7520 外引脚图

具体使用时需要外接集成运算放大器和基准电压源。各引脚情况如下：

D_0～D_9：数据输入端　　I_{OUT1}：电流输出端 1

I_{OUT2}：电流输出端 2　　R_f：10kΩ 反馈电阻引出端

U_{REF}：基准电压输入端　　GND：地

Vcc：电源输入端

AD7520 的主要性能参数有：10 位分辨率；500ns 转换周期（速度）；温度系数是 0.001%/℃；线性误差是±（1/2）LSB（LSB 表示输入数字量最低位）。若用输出电压满刻度范围 FSR 的百分数表示，则为 0.05%FSR。

下面用 AD7520 设计一锯齿波发生器。图 10-9 所示的电路为一个由 10 位二进制加法计数器、AD7520 及集成运放组成的锯齿波发生器。其输出波形见图 10-10。

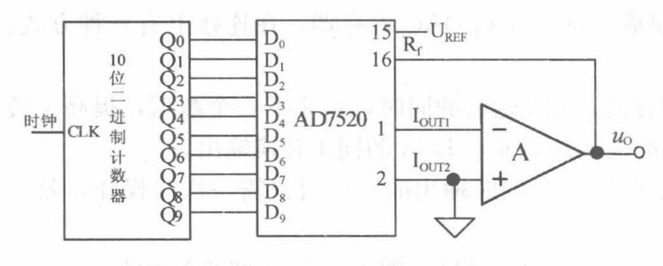


图 10-9　AD7520 组成的锯齿波发生器

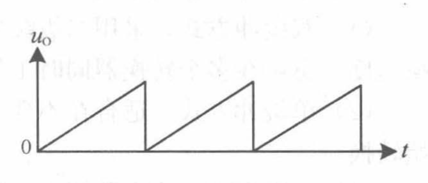


图 10-10　输出的锯齿波波形

10 位二进制加法计数器从全 0 加到全 1，电路的模拟输出电压 u_o 由 0 增加到最大值，此时若再来一个计数脉冲，则计数器的值由全 1 变为全 0，输出电压也从最大值跳变为 0，输出波形又开始一个新的周期。如果计数脉冲连续不断，则可在电路的输出端得到周期性的锯齿波，这可以通过示波器来观察。

3. 串行 DAC 转换器 TLC5615

在后向通道中采用 DAC 转换器是计算机实现以模拟量进行控制的常用方式，而串行 DAC 由于接口电路简单、易于远程操作，以及体积小、功耗低等优点而广泛应用于便携式设备或一些速度要求不高的测控系统中。串行 DAC 的最大缺点是转换速度低。

DIN 1　8 Vcc
SCLK 2　7 OUT
$\overline{CS}$ 3　6 REFIN
DOUT 4　5 AGND
TLC5615

图 10-11　TLC5615 引脚图

TLC5615 的外形结构如图 10-11 所示。器件各引脚含义如下：

DIN：串行数据输入端　　SCLK：串行时钟输入端

$\overline{CS}$：片选信号　　DOUT：串行数据输出端，用于级联

AGND：模拟地　　REFIN：基准电压输入

OUT：DAC 模拟电压输出端　　V_{CC}：电源端

TLC5615 是一种带有 3 线串行接口具有缓冲输入的 10 位 DAC。其输出电压可在 2 倍参考电压 REFIN 的范围内变化，其接口特点是：5V 单电源工作，3 线串行接口，高阻抗基准输入，电压输出可达基准电压的 2 倍，具有内部复位功能。

10.3 A/D转换器

本节介绍A/D转换原理、基本结构、A/D转换器的种类，以及A/D转换器的主要技术参数。最后介绍常用的集成ADC0809的结构原理及应用方法。

10.3.1 A/D工作原理

将模拟量向数字量转换的过程分为两步，第一步是先使用传感器将被测对象产生的连续变化的物理量转换为模拟电信号；第二步再由A/D转换器把模拟信号转换为数字信号。为了将时间与幅值都连续的模拟信号转换成时间与幅值都离散的数字信号，A/D转换需要经过四个阶段，即采样、保持、量化、编码。通常，采样和保持用采样保持电路来完成，而量化和编码则在转换过程中实现。

1. 采样-保持

采样就是将时间上连续变化的信号转换为时间上离散的信号，即将时间上连续变化的模拟量转换为一系列等间隔的脉冲，脉冲的幅度取决于输入模拟量的幅度大小。根据采样定理，采样频率 f_s 必须大于等于输入模拟信号包含的最高频率 f_{max} 的两倍。采样过程操作示意图如图10-12所示。由于A/D转换需要一定的时间，为了保持转换期间输入信号的恒定，采样后的值必须保持不变。采样和保持操作的结果是近似输入模拟信号的“阶梯状”的波形，如图10-13所示。

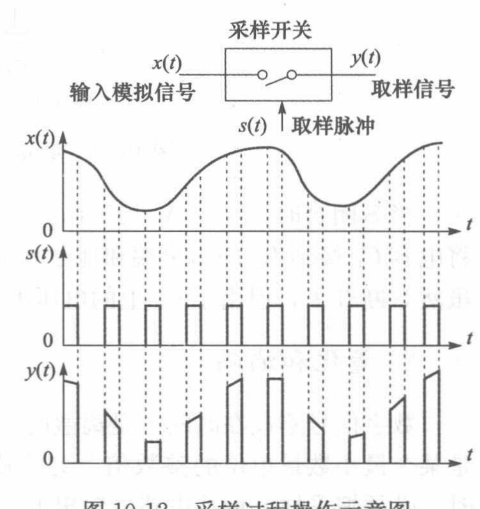
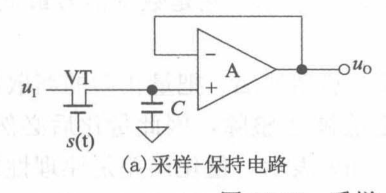
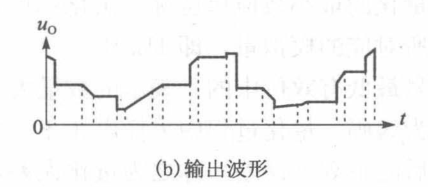


图10-12 采样过程操作示意图



(a)采样-保持电路



(b)输出波形

图10-13 采样-保持电路及输出波形

如图10-8所示，根据取样定理，设取样脉冲 $s(t)$ 的频率为 f_s，输入模拟信号 $x(t)$ 的最高频率分量的频率为 f_{max}，必须满足的条件是：$f_s \geqslant 2f_{max}$。只有这样，图10-12的被采样获得的离散信号 $y(t)$ 才可能正确地反映输入信号大小，并因此能不失真地恢复出原模拟信号。为此，通常取 $f_s=(2.5\sim3)f_{max}$。

若输入信号变化较快，而A/D转换速度较慢，则需要加上采样-保持电路。由于A/D转换需要一定的时间，在每次采样以后，需要把采样的电压保持一段时间。如图10-13

(a) 所示，在取样脉冲 $s(t)$ 有效期间，开关管 VT 导通，u_I 向 C 充电，$u_O(=u_C)$跟随 u_I 的变化而变化；而在 $s(t)$ 无效期间，开关管 VT 截止，$u_O(=u_C)$保持不变，直到下次采样。考虑到集成运放 A 具有很高的输入阻抗，在保持阶段电容 C 上所存电荷不易泄放。

图 10-14 是常用的集成采样-保持器 LF198/298/398 的电路原理图及符号。图中 A_1、A_2 是两个运算放大器，S是电子开关，L是逻辑控制电路。当逻辑输入 v_L 为 1，即 v_L 为高电平时，S闭合，进入采样阶段；v_L 为 0，即低电平时，S断开，进入保持阶段。

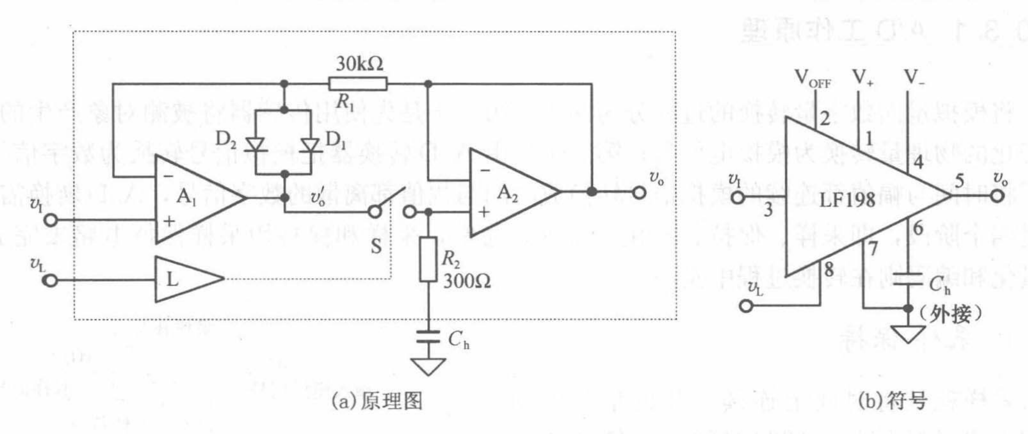


图 10-14 集成采样-保持器 LF198 的电路原理图

当S闭合时，A_1、A_2 均工作在单位增益的电压跟随器状态，所以 $v_o=v_o'=v_I$。如果将电容 C_h 接到 R_2 的引出端和地之间，则电容上的电压也等于 v_I。当 v_L 返回低电平以后，虽然S断开了，但由于 C_h 上的电压不变，所以输出电压 v_o 的数值得以保持下来。

2. 量化和编码

数字信号不仅在时间上是离散的，而且在幅值上也是不连续的。任何一个数字量只能是某个最小数量单位的整数倍。为将模拟信号转换为数字量，在用数字量表示取样电压时，必须把采样一保持电路的输出电压化成这个最小数量单位的整倍数，这个转化过程就叫做量化。量化的最小数值单位称为量化单位，用 Δ 表示。它是数字信号最低位为 1，其他位为 0 时所对应的模拟量，即 1LSB。

数字信号最低有效位中的 1 表示的数量大小，就等于 Δ。把量化后的离散量用二进制数表示，称为编码。量化过程中采样电压不一定能被 Δ 整除，因此量化后必然存在误差。这种量化前后的不等（误差）称之为量化误差，用 ε 表示。量化误差是原理性误差，只能增加二进制位来缩小量化误差。量化级分得越多（n 越大），量化误差就越小。

10.3.2 A/D 转换器工作原理

A/D 转换器有不同的类型。按照转换方法的不同主要分为三种：并联比较型，特点是转换速度快，但精度不高；双积分型，特点是精度较高，抗干扰能力强，但转换速度慢；逐次逼近型，特点是转换精度高。

按工作原理不同，A/D转换器可以分为直接型ADC和间接型ADC。直接型ADC可直接将模拟信号转换成数字信号，这类转换器工作速度快。并行比较型和逐次比较型ADC属于这一类。而间接型A/D转换器先将模拟信号转换成中间量（如时间、频率等），然后再将中间量转换成数字信号，转换速度比较慢。双积分型A/D转换器、计数式A/D转换器和电压/频率转换器等则属于间接型A/D转换器。

1. 并行比较型A/D转换器

图10-15给出了并行比较型A/D转换器的结构框图。此型A/D转换器由2^n-1个模拟比较器、一个2^n-1位寄存器、一个优先编码器和能产生2^n-1个基准电压的2^n个精密电阻组成。注意，图中精密电阻构成的分压电路並未画出，仅标出了比较器基准电压。

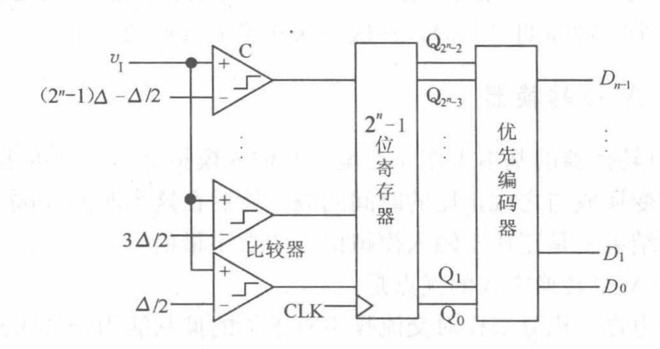


图10-15 并行比较型A/D转换器原理框图

输入模拟电压v_I与各比较器参考电平比较，产生的2^n-1位二进制码，通过寄存器寄存，被译码成n位二进制数（D_0～D_{n-1}），完成模拟信号到数字信号的转换。此类ADC的优点是转换速度快，但输出位数增加一位，所需的电路元件成倍增加，成本较高。

2. 反馈比较型A/D转换器

反馈比较型A/D转换器的基本原理是，由计数器产生一个二进制数，经过D/A转换器将该二进制数转换成模拟电压，此模拟电压和输入的待转换的模拟电压分别送到模拟比较器进行电压比较，根据比较结果控制计数器的二进制数，以逼近输入模拟电压来完成A/D转换，最后获得的计数器中二进制数即为A/D转换后的数字输出。

逐次比较型A/D转换器和计数型A/D转换器都属于反馈比较型A/D转换器，它们内部的原理框图分别如图10-16和图10-17所示。逐次比较型A/D转换器是在计数型A/D转换器基础上用寄存器和控制逻辑电路取代计数器而成。逐次比较型用最快的方法逼近输入的模拟量，而计数型则用计数器递增方式逼近模拟量。显然，逐次比较型A/D转换器转换速度优于反馈比较型A/D转换器。

逐次比较型A/D转换器开始转换时，计数器最高位为1，D/A转换器输出的$v_A=1/2$最大输出电压与输入电压v_I进行比较，若v_A大于v_I则于下个CP脉冲后，计数器高位为0，本位为1；若v_A小于v_I则CP脉冲来到后，计数器高位保持而本位为1。也即第二个

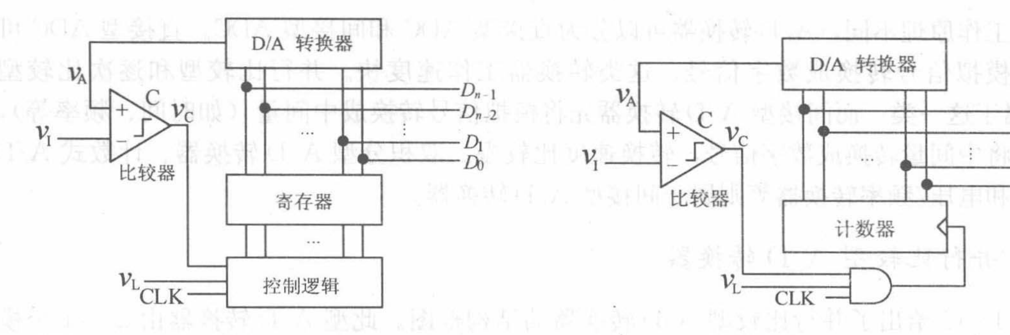


图 10-16　逐次比较型 A/D 转换器原理框图　　图 10-17　计数比较型 A/D 转换器原理框图

CP 后 $v_A = v_{Amax}/4$ 或 $3v_{Amax}/4$；依次类推，最终计数器各位数值被确定。确定 n 位计数器各位值至少需要 n 个时钟周期（T_{CP}），一般一次转换需（$n+2$）个 T_{CP}。

3. 双积分型 A/D 转换器

双积分型 A/D 转换器的基本工作原理是，对输入模拟电压和基准电压分别进行积分，将输入电压平均值变换成与之成正比的时间间隔，然后在这个时间间隔里对固定频率的时钟脉冲计数，计数结果就是正比于输入模拟信号的数字量信号。

基于双积分型 A/D 转换方式的优点是：

(1) 抗干扰能力强。积分采样对交流噪声有很强的抑制能力；如果选择采样时间的整数倍时，则可有效地抑制工频干扰（即来自家庭或环境功率用电源的干扰）。

(2) 工作稳定性好，可实现高精度转换。由于在转换过程中通过两次积分把输入模拟电压和基准电压之比变成了两次计数值之比，故转换结果和精度与积分 R、C 无关。

然而，这种转换方式的缺点是转换速度慢。通常，每秒钟一般只能转换几次到十几次。因此它多用于精度要求高、抗干扰能力强而转换速度要求不高的场合。

10.3.3 A/D 转换器的主要技术参数

A/D 转换器的主要技术指标也有 3 个，即分辨率、转换误差和转换时间。

1. 分辨率

A/D 转换器的分辨率用输出二进制数的位数 n 来表示，位数越多，对输入模拟信号的分辨能力就越强。例如，输入模拟电压的变化范围为 0～5V，输出 8 位二进制数可以分辨的最小输入模拟电压为 $5V \times 2^{-8} = 20mV$；而输出 12 位二进制数可以分辨的最小输入模拟电压为 $5V \times 2^{-12} \approx 1.22mV$。

2. 转换误差

转换误差是指 A/D 转换器实际输出的数字量和理论上的输出数字量之间的差别。常用最低有效位（LSB）的倍数表示。例如某 ADC 的相对精度为±(1/2) LSB，这说明理论

上应输出的数字量与实际输出的数字量之间的误差不大于最低位的 1/2。

3. 转换时间

转换时间指完成一次转换所需的时间，即指从接到转换控制信号开始，直至输出端得到稳定的数字输出信号所经过的时间。显然，转换时间是 A/D 转换速率的重要标志。双积分 ADC 的转换时间在几十毫秒（ms）至几百毫秒之间；逐次比较型 ADC 的转换时间大都可达 10～50μs 之间；并行比较型 ADC 的转换时间可达 10ns。

10.3.4 典型集成 A/D 转换器及应用

在实际测控中，测控系统所能达到的精度和速度最终是由 A/D 和 D/A 转换器的转换速度和转换精度决定的。选用 A/D 转换器可以参考以下四个基本准则：

（1）根据检测通道的总误差和分辨率要求，选取 A/D 转换精度和分辨率。

（2）根据被测信号的变化率及转换精度要求确定 A/D 转换器的转换速率。

（3）根据环境条件选择 A/D 芯片的环境参数。

（4）根据接口设计是否简便及价格成本等选取适当的 A/D 芯片。

为了方便说明 ADC 的使用方法，以下介绍两种比较常用的 ADC 集成器件。

1. ADC0809 的应用

ADC0809 是采用 CMOS 工艺制成的单片 8 位 8 通道逐次比较型 A/D 转换器。器件的核心部分是 8 位 A/D 转换模块，它由模拟比较器、逐次逼近寄存器、开关树、256R 电阻网络及控制和定时等部件构成。其原理框图如图 10-18 所示。

对照 ADC0809 芯片的原理框图和引脚图（图 10-19），各端口功能说明如下：

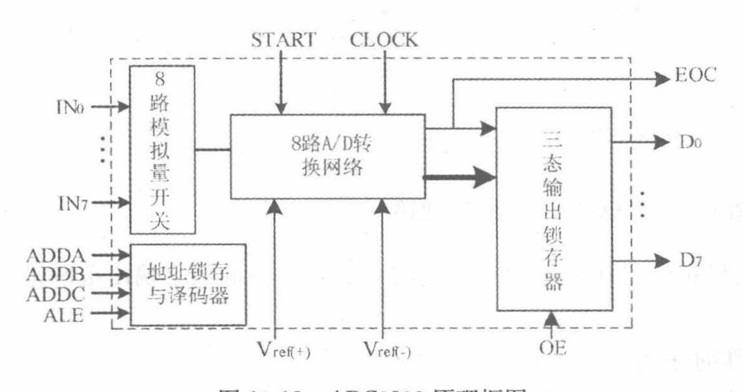


图 10-18 ADC0809 原理框图

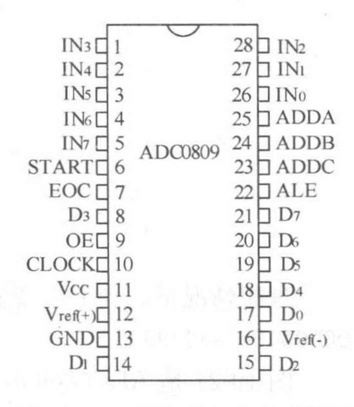


图 10-19 ADC0809 芯片引脚图

- IN_0～IN_7 是 8 路模拟信号输入端。
- ADDC、ADDB 和 ADDA 是 8 路模拟信号的地址码输入端。
- ALE 是地址锁存允许输入信号，上升沿有效，此时锁存地址码便选通相应的模拟信号通道，以便进行 A/D 转换。

- START是启动信号输入端，应在此脚施加正脉冲，当上升沿到达时，内部逐次逼近寄存器复位，在下降沿到达后即启动A/D转换进程。
- EOC是转换周期指示信号，约在START信号上升沿之后1～8个时钟周期内，EOC信号变为低电平，表示已经启动A/D转换；当A/D转换结束后，EOC变为高电平，指示A/D转换结束。转换结束后，转换好的数据可以通过数据端口读出。
- OE是输出允许信号，高电平有效；低电平时ADC数据口呈高阻态。
- CLK是时钟信号输入端，外接时钟频率最高为640kHz。
- V_{CC}是+5V单电源输入端。
- $V_{REF(+)}$、$V_{REF(-)}$分别是基准电压的正端和负端。一般$V_{REF(+)}$接+5V（假设电源端所接的+5V具有良好的稳定性），$V_{REF(-)}$接地。
- D_7～D_0是数字信号输出端。

ADC0809由ADDC、ADDB和ADDA三个地址输入端，选通8路模拟输入通道的任意一路进行A/D转换，地址输入端与模拟输入通道的选通关系如表10-1所示。

表10-1 地址输入与模拟输入通道的选通关系

选通模拟通道		IN_0	IN_1	IN_2	IN_3	IN_4	IN_5	IN_6	IN_7
地址	ADDC	0	0	0	0	1	1	1	1
	ADDB	0	0	1	1	0	0	1	1
	ADDA	0	1	0	1	0	1	0	1

图10-20是ADC0809的一个典型应用电路。待测模拟信号v_i被送入输入端IN_0，转换结果由D_7～D_0输出，CLOCK时钟脉冲约500kHz；ADDC、ADDB、ADDA地址端取000。在启动端START上加一正单次脉冲，ADC0809即开始A/D转换。

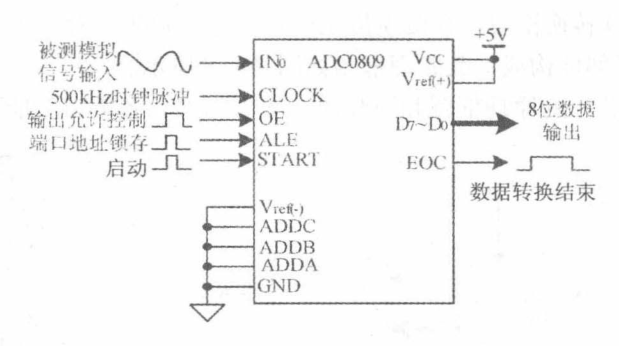


图10-20 ADC0809典型应用电路

理想情况下，当IN_0端输入模拟信号在0～5V范围内时，其转换后的数字输出范围是00000000～11111111。

图10-21是ADC0809的工作时序图。

根据时序图，ADC0809完整的采样转换工作过程可归纳为如下四步：

(1) 当模拟量送入通道IN_0（或IN_i）后，控制电路将标志该通道编码的ADDC、ADDB、ADDA地址信号输入到地址寄存器，由地址锁存ALE锁存这三位地址信号。

(2) 启动转换命令START产生一个上升脉冲后，启动A/D转换。

(3) 启动转换开始后，以及在转换过程中，EOC变为低电平；转换结束后，EOC变

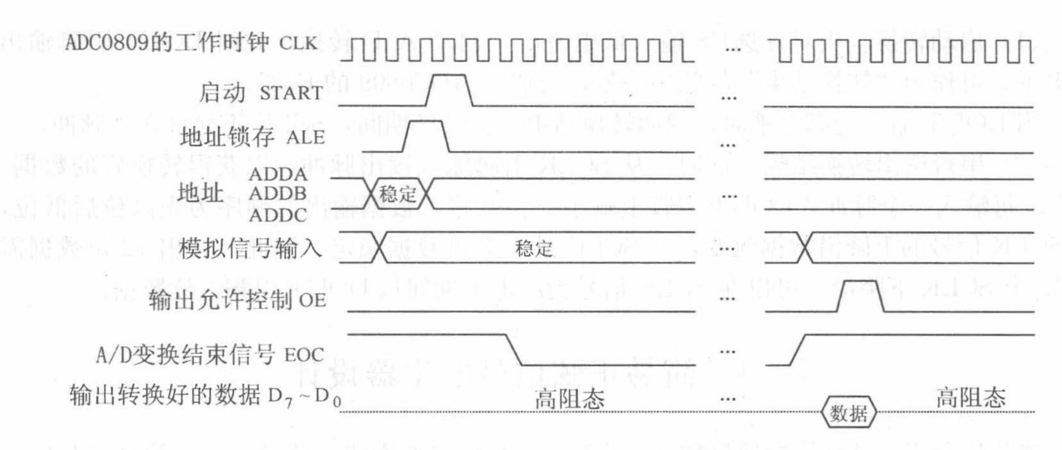


图 10-21　ADC0809 工作时序

为高电平。EOC 可作为计算机的中断请求信号，或作为其转换控制状态机的状态译码器的输入控制信号，即由 EOC 决定状态机下一状态的转向。

(4) 转换结束后，由控制电路向 ADC0809 的 OE 输出高电平，即打开三态缓冲器把转换好的结果 $D_7 \sim D_0$ 输出。至此，一次 A/D 转换便完成了。

【例 10-2】 对于图 10-20 所示的应用电路，ADC0809 的输入模拟电压满量程为 5V，当输入电压为 1.96V 时，求对应的输出数字量。

解：输入模拟电压与输出数字量对应的十进制数成正比：$V_i = K \cdot (D)_{10}$，因此，$\frac{5}{(11111111)_2} = \frac{1.96}{(D)_{10}}$；得输出数字量 $(D)_{10} \approx 100 = (01100100)_2$。

2. 串行 A/D 转换器 MAXIM187/189

前面曾提到，A/D 转换器有并行和串行两种数据输出形式。虽然并行 ADC 数据传输速度快，但有引脚多、体积大、占用控制器接口多等缺点；而目前常用的许多类型串行 ADC 的传输速率已有了很大的提高，并且具有体积小、功耗低、占用控制器接口少的优点。因此，串行 ADC 的应用已越来越广泛。

串行 A/D 转换器的生产厂商较多，著名的厂商有：ADI、NS（国家半导体）、TI（德州仪器）、MAXIM 等。

由于串行 A/D 转换器的基本结构功能相似。串行 A/D 转换芯片的型号也较多。例如 MAXl71 是带光电隔离的 12 位串行 A/D 芯片；MAXl47 是 8 通道的 12 位串行 A/D 芯片等。比较典型的是 MAXl87/189，此款器件是 MAXIM 公司的具有 SPI（Serial Peripheral Interface）总线接口的 12 位逐次逼近型 A/D 转换芯片，DIP8 封装，外接元件简单，使用方便。具有一个模拟量通道，单一＋5V 供电，转换速度和时间分别为 75kHz 和 8.5μs；内建采样/保持器（Track/Hold），可转换 0～5V 模拟电压。

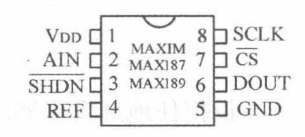


图 10-22　MAX 187/189 引脚图

MAXl87/189 芯片引脚如图 10-22 所示，其 A/D 转换主要步骤如下：

(1) 启动转换。当对片选 $\overline{CS}$ 输入低电平时，启动 A/D 转换，此时 DOUT 引脚输出低电平，可作为“转换结束”的状态信号，类似于 ADC0809 的 EOC。

当 DOUT 输出变高电平时，说明转换结束。在转换期间，SCLK 不允许送入脉冲。

(2) 串行读出转换结果。此时，从 SCLK 引脚输入读出脉冲，以获得转换好的数据。SCLK 每输入一个脉冲，DOUT 引脚上输出一位数据，数据输出的顺序为先高位后低位，在 SCLK 信号的下降沿数据改变，在 SCLK 的上升沿数据稳定。因此，读出 12 位数据需要 13 个 SCLK 下降沿。可以在 SCLK 信号为高电平期间从 DOUT 引脚上读数据。

10.4　简易正弦信号发生器设计

本节将介绍一个常用的通过数字方式产生模拟信号的方法，即设计一个简易正弦信号发生器。此设计涉及 ROM 和 DAC 的应用，这里主要使用 DAC0832 和 LPM_ROM 来完成实验。通过这个示例，可以在实际设计中进一步了解 DAC0832 的性能和使用方法，并进一步熟悉 LPM 模块的调用及存储器初始化文件建立的方法。

10.4.1　工作原理

图 10-23 所示的是正弦信号发生器的结构框图，它由四个部分组成：

(1) 计数器或地址发生器（这里选择 6 位），用来作为正弦波数据 ROM 的地址信号发生器。ROM 中的数据将随地址数据的递增而输出波形数据，然后由 DAC 输出波形。

(2) 正弦信号数据 ROM（选择 6 位地址线，8 位数据线），含有 64 个 8 位数据（正好一个正弦波周期）。

(3) 原理图顶层设计。这个可参考第 6 章和第 8 章。

(4) 8 位 D/A（可使用 DAC0832）。DAC 的输出接示波器。

图 10-23 所示的信号发生器原路框图中，顶层文件（即工程文件）在 FPGA 中实现，包含两个部分：波形数据 ROM 的地址信号发生器，这由 6 位计数器担任；正弦数据 ROM，可用 LPM_ROM 模块实现。

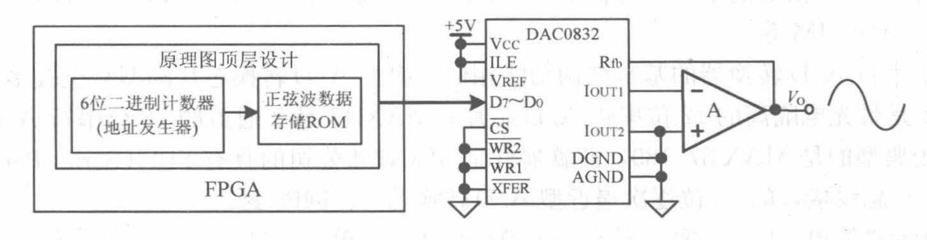


图 10-23　正弦信号发生器结构框图

设担任地址发生器的二进制计数器的时钟 CLK 的输入频率为 f_0，而 8 位输出数据的 ROM 中存放的一个周期的正弦波数据有 64 个点，即 ROM 有 64 个 8 位二进制数据。那么最后由 D/A 输出的、可以在示波器上看到的正弦波的频率 f 应该是：$f=f_0/64$。

10.4.2 定制初始化波形数据文件

在此必须首先确定图 10-23 中 ROM 内的波形数据文件。这里以 64 点正弦波形数据为例说明。关于 mif 类型的存储器初始化文件的格式在第 9 章中已经作过讨论，获得此文件的途径有多种。

这里介绍直接通过 Quartus Ⅱ中选择 LPM_ROM 数据文件编辑窗来建立：在 File 菜单中选择 New（图 6-3），并在 New 窗中选择 Memory Files 项，再选择 Memory Initialization File 项，单击 OK 按钮后产生 ROM 数据文件大小选择窗（图 10-24）。根据 64 点 8 位正弦数据的情况，可选 ROM 的数据数 Number 为 64，数据宽 Word size 取 8 位。单击 OK 按钮，将出现如图 10-25 所示的空的 mif 数据表格，表格中的数据格式可通过鼠标右键点击窗口边缘的地址数据弹出的窗口选择。然后将波形数据填入此表中，完成后如图 10-26所示。此表中任一数据（如第三行的 99）对应的地址为左列与顶行数之和（如 16+2=18，十六进制为 12H，即 00010010）。在 File 菜单中单击 Save as 按钮，保存此数据文件，在这里不妨取名为 romd. mif。

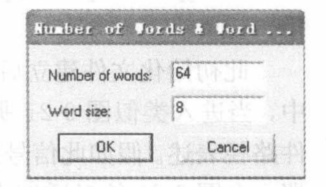


图 10-24 选择 mif 文件格式

图 10-25 为 mif 文件输入数据

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	254	252	249	245	239	233	225
8	217	207	197	186	174	162	150	137
16	124	112	99	87	75	64	53	43
24	34	26	19	13	8	4	1	0
32	0	1	4	8	13	19	26	34
40	43	53	64	75	87	99	112	124
48	137	150	162	174	186	197	207	217
56	225	233	239	245	249	252	254	255

图 10-26 完成正弦波信号数据输入

用这种方法建立 mif 文件的好处是直观，方便；缺点是当数据量大时，十分费时间，且容易出错，例如在最后一章介绍的 DDS 信号发生器中 ROM 的 mif 文件生成就不能用这种方法来完成。所以要用第 9 章中介绍的方法来编辑 mif 文件，或用更方便的方法，就是直接根据附录介绍的 mif 文件生成软件来实现。

当前示例的 romd. mif 文件格式如例 10-3 所示。其中地址和数据都为十六进制，冒号左边是地址值，右边是对应的数据，并以分号结尾。注意每一组数据都必须排一行！

【例 10-3】 romd. mif 文件。

```
WIDTH=8;
DEPTH=64;
ADDRESS_RADIX=HEX;
DATA_RADIX=HEX;
CONTENT BEGIN
```

```
    0      :      FF;
    1      :      FE;
    2      :      FC;
    3      :      F9;
    4      :      F5;
     …(数据略去)
    3D     :      FC;
    3E     :      FE;
    3F     :      FF;
END;
```

此初始化文件建立后，先存于当前工程的文件夹中。此外，在调用 LPM_ROM 过程中，当进入类似图 9-21 所示对话框时，选择此初始化文件 romd. mif 时应该注意准确的文件路径表述。假如此信号发生器工程的主文件存放路径和文件 romd. mif 都是 d:\sin_gnt\，那么在图 9-21 的对话框中的 File name 栏，可以表述此文件路径的方式有两种：d:\sin_gnt\romd. mif 或直接 romd. mif。对此，推荐后一种表达方式。因为前一种方式已固定了路径指向，当此工程被复制到其他地方时，Quartus Ⅱ的编译就有可能找不到 romd. mif 文件了。又假如文件 romd. mif 的存放路径是 d:\sin_gnt\dat\romd. mif，那么在图 9-21 的对话框中的 File name 栏，可以表述此文件路径的方式也有两种：d:\sin_gnt\dat\romd. mif 或 . /dat/romd. mif。同理，推荐使用后一种表达方式，因为这里的点斜杠"./"默认代表了当前工程的路径。

10. 4. 3 定制 LPM ROM 元件

在设计正弦信号发生器前，必须首先完成存放波形数据 ROM 的设计。

设工程名是 singt，路径是 D:\sin_gnt\。原理图顶层设计如图 10-27 所示。CLR 是计数器清 0 信号，高电平有效；CNT6B 模块是 LPM 的 6 位计数器模块，做地址发生器；DAC_OUT[7..0] 是 ROM 数据输出，接外部 DAC；CNT6B[5..0] 可在仿真波形图中看到，如果在示波器中观察，一定是三角波。

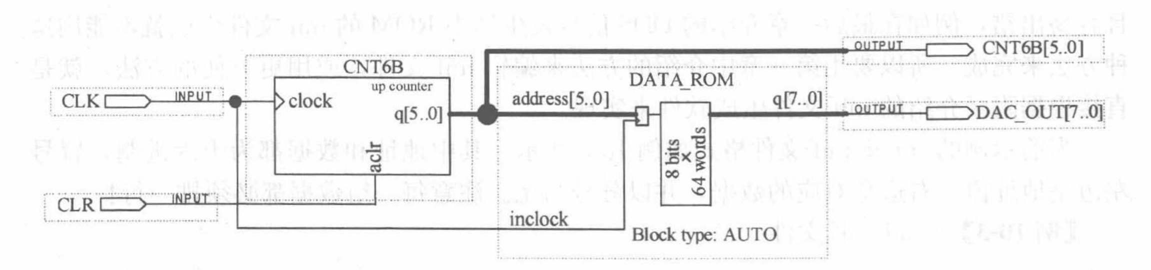


图 10-27 简易正弦信号发生器顶层电路设计

图 10-27 所示的两个模块的参数设置可参考第 9 章。其中选择 ROM 控制线是异步清 0 aclr；地址线位宽和 ROM 中数据数分别为 6 和 64；选择地址锁存 inclock。

10.4.4 完成顶层设计

最后按照图 10-27 把电路图连接好。编译后首先进行时序仿真。除确定设计系统设计无误外，还必须确定 ROM 中的初始化文件数据是否已被加载于 LPM_ROM 中。

设计流程主要包括编辑顶层设计文件、创建工程、全程编译、时序仿真、了解时序分析结果、引脚锁定、再次编译并下载、对 FPGA 的存储单元在系统读写测试等。

图 10-28 是仿真结果。波形显示，随着每一个时钟上升沿的到来，输出端口将正弦波数据依次输出。将这些数据与图 10-26 的 romd. mif 文件的数据比较，可以看出，设计结果是正确的。DAC_OUT 输出数据类型是无符号十进制类型：Unsigned Decimal。

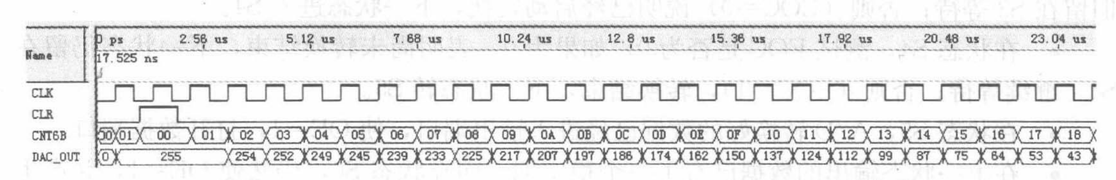


图 10-28 当前工程仿真波形输出

最后是硬件测试。首先根据实验系统的 FPGA 的接口情况锁定引脚再编译，最后下载 singt. sof 后，打开±12V 电压开关。考虑到 DAC0832 的数据建立速度是 1μs，所以 CLK 的频率不宜太高。DAC0832 的输出波形信号可以通过示波器观察。

为了便于对下载于 FPGA 中的 ROM 中数据的了解和编辑，可以利用 Quartus Ⅱ 的 In-System Memory Content Editor 来实现（使用方法可参考文献 [1]）。

10.5 A/D 采样控制电路设计

使用 ADC 对模拟信号进行采样，并把采样后所得的二进制数据存入指定的寄存器或存储器中，必须有一个针对特定 ADC 的合适的控制电路。可以有多种方式对 ADC 的采样进行控制，如用微型计算机、单片机或用数字电路构成的状态机。相比之下，在控制速度、可靠性、成本等多方面，用状态机控制 ADC 采样有更多的优势。

为了便于实验和说明，以下以 ADC0809 为例，说明 ADC 采样控制状态机的设计方法，希望读者能举一反三，设计出控制其他实用系统的不同状态机来。

10.5.1 控制原理

用状态机对 ADC0809 进行采样控制首先必须了解其工作时序，然后据此作出状态图，最后设计出控制电路。这可以参考 10.3.4 节及时序图（图 10-21）和端口信号图（图 10-19）。

需要注意的是，ADC0809 工作时需要一个工作时钟，即图 10-21 的 CLK，一般约几百千赫，而控制 ADC0809 的状态机也必须有一个工作时钟，即状态机工作时钟。这个时钟的频率可以比较高，最高可以有数十兆赫。由于状态机的工作时钟比 0809 的工作时钟

高得多，所以，需要当启动 0809 转换信号 START 有效后，首先测试转换状态标志信号 EOC 是否从高电平变为低电平了；当 EOC=0 后，再测试 EOC 从低电平回到高电平的变化时刻，此时才是 A/D 转换结束的标志！

根据这些考虑，可以画出状态机对 0809 的控制状态图，如图 10-29 所示。由状态图可以看到：

- 首先在初始态 S0 对 0809 进行初始化，然后无条件进入状态 S1。
- 在状态 S1 中将模拟信号通道地址锁入寄存器，即在此状态使 ALE 产生一个上升沿。这里设（ADDC，ADDB，ADDA）=000，即选择模拟信号通道是 IN0。
- 在状态 S2 启动转换（START=1），之后进入 S3 状态。
- 在状态 S3，测试 EOC 是否为 1，如果为 1，说明 0809 尚未启动转换，下一状态仍旧留在 S3 等待；否则（EOC=0）说明已经启动转换，下一状态进入 S4。
- 在状态 S4，测试 EOC 是否为 0，如果为 0，表明尚未转换结束，下--状态仍留在 S4，继续等待；否则（EOC=1），转换结束，下一状态转 S5。
- 在状态 S5，A/D 转换好的数据已经进入输出端口，使 OE=1，打开数据端口。
- 在上一状态输出的数据已有了一个稳定期，而在状态 S6，仍然使 OE=1，并产生一个数据锁存信号上升沿（LOCK=1），将转换好的数据存入寄存器中；进入状态 S7，准备返回初始态 S0。

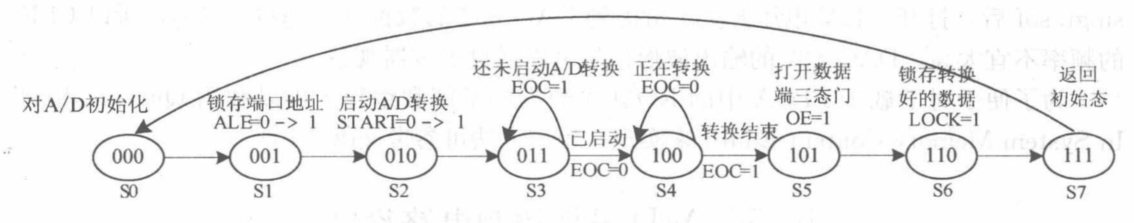


图 10-29 ADC0809 采样控制状态图

这个状态机的时序元件可以用 3 个 D 触发器构成。那么它们最多可以有 8 个状态，恰好代表 S0~S7。这样就不用担心自启动问题了，因为在任何时刻，此状态机都是处于合法状态中（实际上，最多只需要 5 个状态就能完成控制）。

10.5.2 ADC 采样控制电路设计

根据 ADC0809 的接口特性和控制状态机的要求，其采样控制的状态机电路如图 10-30 所示，此状态机结构与图 8-17 所示的模型相同。图 10-30 中的元件 AD_SDCD、DFF3、AD_CDCD 和 74374b 构成了一个控制 0809 进行 A/D 转换的状态机。

AD_SDCD 是状态译码器，它根据现态状态编码 C[2..0] 和来自 0809 的 A/D 转换状态信息 EOC，决定状态的走向；AD_CDCD 是控制译码器，负责向 0809 输出控制信号 ALE、START、OE、LOCK；DFF3 是 3 个 D 触发器组成的状态寄存器，或者称状态驱动器，属于纯时序电路。这个状态机的驱动时钟，或称工作时钟是接在 DFF3 模块 CLK 端的，它决定了状态机的工作速度。

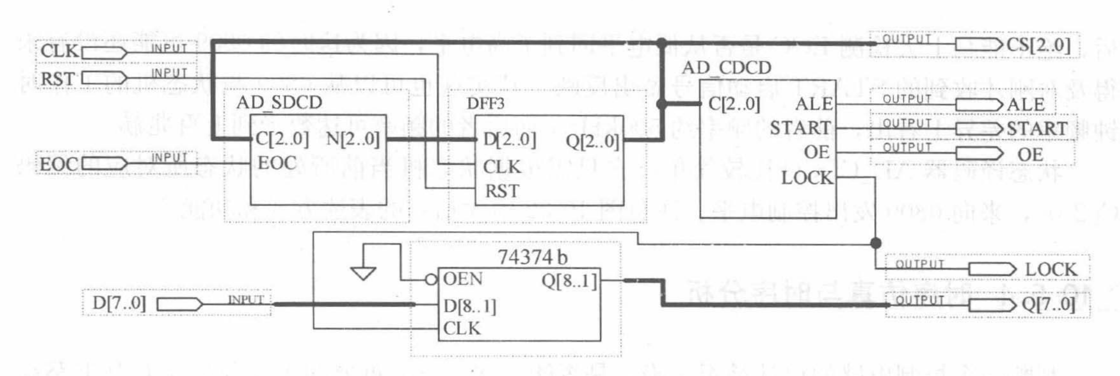


图 10-30 ADC0809 采样状态机控制电路

AD_SDCD 和 AD_CDCD 模块的 Verilog 表述分别如图 10-31 和图 10-32 所示。其中 AD_SDCD 输出的数据 N[2..0] 是次态状态码，输入的数据 C[2..0] 是现态状态码。AD_CDCD 输出的 LOCK 控制寄存器 74374，它负责锁存 ADC0809 采样获得的数据，它的输入口 D[7..0] 直接与 ADC0809 的数据口相接。

```
1  module AD_SDCD (C,N,EOC );
2     input[2:0] C ;   input EOC;   output[2:0] N ;
3     reg[2:0] N ;
4     always @(C or EOC)  begin
5       case ( C )
6         3'B000 : N <= 3'B001 ;
7         3'B001 : N <= 3'B010 ;
8         3'B010 : N <= 3'B011 ;
9         3'B011 : if (EOC==1'b0) N<= 3'B100; else  N<=3'B011 ;
10        3'B100 : if (EOC==1'b1) N<= 3'B101; else  N<=3'B100 ;
11        3'B101 : N <= 3'B110 ;
12        3'B110 : N <= 3'B111 ;
13        3'B111 : N <= 3'B000 ;
14      default : N <= 3'B000 ;
15        endcase
16     end
17  endmodule
```

图 10-31 状态译码器 AD _ SDCD 的描述

```
1  module AD_CDCD (C,ALE,START, OE,LOCK );
2     input[2:0] C;  output ALE, START,OE,LOCK;
3     reg ALE, START, OE, LOCK ;
4     always @(C )  begin
5       case ( C )
6         3'B000 : {ALE,START,OE,LOCK}<= 4'B0000;
7         3'B001 : {ALE,START,OE,LOCK}<= 4'B1000;
8         3'B010 : {ALE,START,OE,LOCK}<= 4'B1100;
9         3'B011 : {ALE,START,OE,LOCK}<= 4'B1000;
10        3'B100 : {ALE,START,OE,LOCK}<= 4'B0000;
11        3'B101 : {ALE,START,OE,LOCK}<= 4'B0010;
12        3'B110 : {ALE,START,OE,LOCK}<= 4'B0011;
13        3'B111 : {ALE,START,OE,LOCK}<= 4'B0000;
14      default : {ALE,START,OE,LOCK}<= 4'B0000;
15        endcase
16     end
17  endmodule
```

图 10-32 控制译码器 AD _ CDCD 的描述

10.5.3 广义译码器设计

对于图 10-30 所示的状态机电路，状态译码器和控制译码器的设计是关键。

对于状态译码器 AD_SDCD 的设计，必须准确了解和把握状态机控制对象的时序特点和工作特性。本项设计的对象是 ADC0809，所以可以根据其采样控制时序图和确定下来的状态图（图 10-29），描述对应的 case 语句。对于此模块的描述要把握好对来自 0809 的转换状态信号 EOC 的监测，因为它决定了状态 3 和状态 4 的转向。此外要注意，在图 10-31 的程序中用了两条 if 语句，第一条检测 0809 是否进入 A/D 的转换进程；第二条检测 A/D 转换是否结束。其实，若使用传统的单片微机来控制 0809，程序中仅有对应第二条 if 语句描述的测试。这是因为，单片微机的运行速度比状态机慢得多，当它向 0809 发出启动转换命令后，再来检测 EOC 时，已经过去许多时间了；而这段时间中，0809 输出给单片机的 EOC 肯定早已从高电平变成低电平了，无需再检测这个变化了。所以，这时单片机只需检测 EOC 是否变成 1 就够了，即检测 A/D 转换是否结束。

然而，对于状态机却不同，状态机的运行速度很高，当其通过 START 发出启动信号

后，绝不能马上去检测 EOC 是否从低电平回到了高电平，因为这时的 0809 可能还没有来得及对刚才收到的 START 启动信号做出反映。其实这也可以从 0809 与状态机的工作时钟频率的差异上看出，前者的频率约 500kHz，而后者的频率可达数十到上百兆赫。

状态译码器 AD_CDCD 比较简单，它只需根据状态机当前所处的状态所对应的编码 C[2..0]，来向 0809 发出控制电平。注意图 10-32 中大括号的表述方式和功能。

10.5.4 时序仿真与时序分析

判断一个控制电路的设计是否正确，是否能正常工作，重要的不是仅仅了解其电路结构、组成和元件的语句描述，而是严格的时序仿真。因此，接下去应该对电路图 10-30 进行编译和时序仿真。从仿真波形上了解和判断设计的可行性和合理性。

在编辑仿真文件时，特别注意激励信号设置的合理性。图 10-33 是仿真报告波形图，其中的激励信号，如 CLK（状态机工作时钟）、RST、EOC、输出数据 D[7..0] 的波形设计都是参考 0809 的时序图的。波形表明，此项设计符合最初的要求：

图中的 CS 是现态状态指示，当 CS=0 时是初始态 S0；在进入状态 S1（CS=1）时，ALE 出现一个上升沿，将 0809 的地址锁存进地址寄存器；在进入状态 S2（CS=2）时，START 出现一个上升沿，启动 0809 转换工作；在状态 S3 的整个过程中（CS=3），由于 EOC=1，所以一直停留在状态 S3；当 EOC 一变成 0，在接下来的第一个时钟上升沿处，即刻进入状态 S4（CS=4），但由于一直有 EOC=0，所以一直停留在本状态 S4。直到 EOC 由 0 变成 1 时，紧接着 CLK 的一个上升沿的到来，即进入状态 S5（CS=5）。由于此时转换数据已经完成，所以在此打开 OE 控制的三态门。

为了稳定 0809 的输出数据，直到进入状态 S6 后，才安排 LOCK 出现一个上升沿，将 0809 数据口的数据 A9 锁入 74374 中，而在之前的“ZZ”表示转换未结束，OE 也未打开，输出为高阻态。

时序图的详细情况已标注于图 10-33 右侧了。

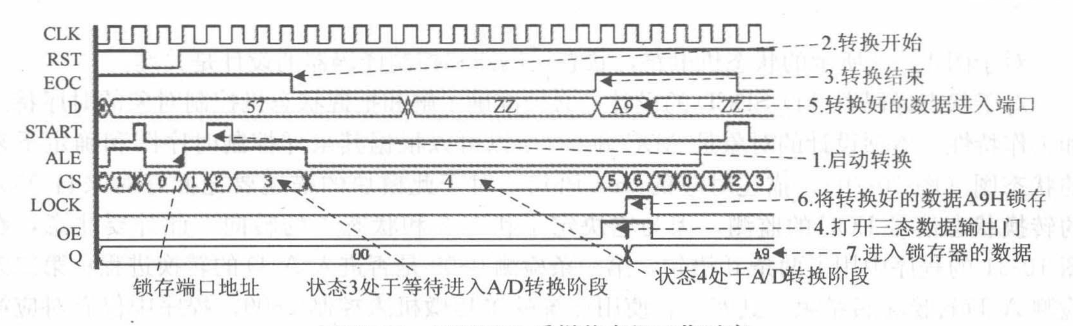


图 10-33　ADC0809 采样状态机工作时序

10.5.5 硬件实现与硬件实测

最后锁定引脚，编译后下载于 FPGA 中后，可以在实验系统上硬件验证此项设计的正

确性。在实验中可以将 74374 的输出用实验系统上的数码管显示，输入的模拟电压可以用实验系统上的电位器产生。如果一切正确，旋转电位器时，可以看到数码管的数值变化。由于电位器的电压变化范围是 0～5V，数码管显示的对应数据应该是 00～FFH。

此项设计是原理验证性的，因为只能分别显示一个点的采样值。为了能对连续模拟信号采样，并保持起来显示和分析，可以将图 10-30 中的 74374 改为 LPM_RAM。

习 题

10-1 常见的 D/A 转换器和 A/D 转换器分别有哪几种类型？其特点分别是什么？

10-2 有一 8 位 T 型电阻网络 D/A 转换器，已知 $V_R=+5V$，$R_f=2R$，试求 $D_7\sim D_0$ 分别为 0000 0000，1000 0000，11111111 时的输出电压 V_O。

10-3 有一 8 位 T 型电阻网络 D/A 转换器，已知 $V_R=-5V$，$R_f=R$，试求 $D_7\sim D_0$ 分别为 00000000，10000000，11111111 时的输出电压 V_O。

10-4 图 10-4 所示电路中，$V_R=10V$，求对应二进制数码的输出电压值，写出 V_O 的逻辑表达式。

10-5 若要求 D/A 转换器的精度误差小于 1%，则 D/A 转换器至少需要用多少位？

10-6 在图 10-4 所示电路中，$n=4$，$V_R=10V$，$R=10k\Omega$，求当 $D_7\sim D_0=01001000$ 时的输出电压值 V_O，当 $D_7\sim D_0=11001010$ 时的输出电压值 V_O。

10-7 在 A/D 转换电路中，为什么要加采样-保持电路？简要说明采样-保持电路的工作原理。保持电容 C_H 对电路有何影响？

10-8 逐次逼近式 A/D 转换器有何特点？主要由哪几部分组成？各部分的基本功能是什么？

10-9 并行比较型 A/D 转换器主要由哪几部分组成？各部分的基本功能是什么？

10-10 简要说明影响 DAC 和 ADC 性能的主要技术参数有哪些？

10-11 某 8 位 A/D 转换器的输入模拟电压满量程为 5V，当输入电压为 1.52V 时，求输出值。

10-12 双积分型 A/D 转换器有何特点？请举例说明。

10-13 如果要求 ADC 对输入电压的分辨率为 2.5mV，其满刻度输出所对应的输入电压为 8.125V，该 ADC 至少应有多少位字长？

10-14 某 12 位 ADC 电路满量程值输入电压为 10V，当输入电压值分别为 75.5mV、5.83V、9.62V 时，输出数字量分别是多少？

实 验

10-1 简易正弦信号发生器设计

（1）按照 10.4 节的流程，设计一个正弦信号发生器。要求 ROM 是 8 位数据线，8 位地址线。256 个 8 位波形数据的 mif 文件通过两种方式建立，一种用 Quartus Ⅱ的专用编辑器建

立，另一种是使用附录的mif文件生成器建立。首先创建原理图工程，调用LPM_ROM等模块；在原理图编辑窗中绘制电路图，全程编译，对设计进行时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译，编程下载于FPGA中，用实验系统上的DAC0832作波形输出，用示波器来观察波形。完成实验报告。

(2) 学习使用Quartus Ⅱ的In-System Memory Content Editor来观察FPGA中LPM_ROM中的波z形数据，并在在线改变数据后，从示波器上观察对应的输出波形的改变情况。

(3) 学习使用Quartus Ⅱ的SignalTap Ⅱ观察FPGA输出的正弦波形。

10-2 8通道逻辑分析仪示波器显示控制电路设计

根据第9章的实验9-2和实验10-1，为8通道逻辑分析仪设计一个显示控制电路。

首先根据实验10-1，设计一个锯齿波信号发生器。此发生器不需要LPM_ROM，只要一个8位计数器（计数时钟频率约60kHz）即可，让此计数器的输出直接接DAC0832，即可产生周期性锯齿波。选择示波器X-Y功能，让输出的锯齿波接示波器的X端，控制横向扫描。

同时，用产生锯齿波的同一时钟同步控制图9-24逻辑分析仪电路中RAM0的时钟inclock；另增加一个8选1多路选择器对ARAM0的8位输出进行选择。多路选择器的输出随着1至8路选择的递增，每一次为输出结果增加一个阶梯增量（做一个译码器类加法器），然后把输出接示波器的Y端，控制纵向幅度。

这样一来，就可以将采样获得的8路脉冲波形数据同时显示在示波器上。使示波器类似于一个8踪示波器，同时展示8路采样所得的脉冲波形。最后完成实验报告。

10-3 A/D转换实验

按照图10-20的ADC0809典型应用电路连接。CLK接10kHz时钟信号，用精密电位器从直流电源分压，得到0～+5V的被测电压，被测电压从运算放大器的输入端V_I输入。接通电源后，在启动端START加一正单次脉冲，即开始A/D转换。调节电位器改变输入电压，列表分别记录A/D转换结果。实验报告要求：画出总体电路图，简述ADC0809的工作原理和调试步骤，根据实验记录的数据，作出A/D转换的特性曲线，分析实验结果，并对调试中遇到的问题进行分析，说明解决方法。

10-4 A/D采样状态机控制电路设计

按照10.5节的流程，设计一个控制A/D采样的状态机。首先创建工程，在原理图编辑窗中绘制电路图，全程编译，对设计进行时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译，编程下载于FPGA中，用实验系统上的FPGA和ADC0908完成实验，待测模拟信号来自电位器，采样所得的2位十六进制数据可以用数码管显示。最后完成实验报告。

10-5 采样保持器实验

按照图10-14进行电路连接。v_I输入连续的模拟信号，v_L接TTL电平的采样脉冲，

C_h 是外接的保持电容。根据香农采样定理，采样脉冲的频率应$\geqslant f_{max}$，才能不失真地恢复被测信号。用示波器观察采样-保持器的输出信号 v_O 的变化情况。要求：

（1）分别观察和记录采样期和保持期输出波形的变化情况。

（2）改变采样脉冲的频率，观察输出波形的变化。

（3）改变保持电容，观察采样期和保持期输出波形的变化情况。

实验报告要求：画出总体电路图，简述采样-保持器工作原理，调试步骤，分析实验结果，并对调试中遇到的问题进行分析，说明解决方法。

思考题1：采样脉冲的频率对输出波形有何影响？

思考题2：保持电容对采样信号的影响：当保持电容很大时，对高频信号的采样有何影响？当保持电容很小时，对低频信号保持输出有何影响？

第11章 脉冲电路及其分析

本章讨论的内容主要是各类振荡器、单稳态电路、施密特触发器和555定时器等模数混合型电路，属于传统的脉冲电路部分。在传统数字电路教材中，这部分内容通常在较前面的章节中出现。但根据本书前言的讨论，这部分内容不但可以放在课程的最后进行，甚至当学时数不够时，可暂时放弃对本章内容的教学，而仅作为课后的自学内容（待模拟电子线路课程的学习结束后，效果会更好），从而使得读者无需太多的预备知识便可尽可能早地进入数字电路的学习和实验阶段。实践证明，这并不会影响整体教学效果。事实上，现在有些教材已将这部分内容收进模拟电子线路课程中了。

对于脉冲电路的研究和应用在早期数字电路设计中具有不可忽视的重要性。脉冲电路主要用于低速小规模数字电路中的脉冲波形发生和波形信号调理，比如振荡器用于时钟脉冲波形的发生；其中的单稳态电路和施密特触发器可以用于输入或输出脉冲波形的规整、叠加的干扰信号的去除；而经典的555定时器既可以用于发生给定周期、给定占空比的脉冲波形，又可以接成单稳态电路和施密特电路用于脉冲波形的处理。

11.1 多谐振荡器

在数字电路中最常用的是时钟脉冲，通常的时钟脉冲都是矩形波脉冲。可以有多种方法来生成连续的矩形波，例如，可以使用多谐振荡器来生成矩形波脉冲。多谐振荡器是一种自激振荡器，可以在电路电源加电后即发生振荡，产生矩形脉冲。由于矩形波中含有丰富的谐波分量，所以把通过自激振荡发生矩形波的振荡器称之为多谐振荡器。

11.1.1 环形多谐振荡器

最简单的多谐振荡器的构成可以使用普通非门元件来实现，其电路连接方式如图11-1（a）所示，在这个电路中，非门完成输入信号到输出信号相位的180度反转。使用奇数个非门使得整个环路处于两个暂稳态之间变化，从而在 f_{out} 处送出矩形波脉冲，如图11-1（b）所示。

这个电路的输出脉冲频率取决于非门的延迟。若设每个非门的延迟是 t_{INV}，那么图11-1（a）电路中，从输出为低电平的暂稳态到输出为高电平的暂稳态需要经历3个 t_{INV}，于是两个暂稳态的时间，即是脉冲周期应该为 $6t_{INV}$。显然，电路输出脉冲频率是 $f_{out}=1/(6t_{INV})$。

如果需要降低输出的脉冲频率，可以使用更多个（奇数个）非门。但这种构成多谐振荡器的方法没有什么实用意义，因为输出频率受器件工艺和工作温度的影响过大。

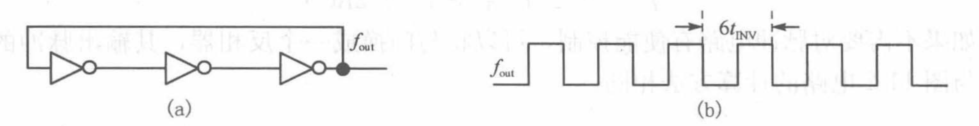


图 11-1 用非门构成的环形振荡器及其输出波形

11.1.2 非对称型多谐振荡器

图 11-2 是一种常用的非对称型多谐振荡器电路。由于采用了 RC 阻容元件作振荡元件，也被称为 RC 振荡器。该电路由一个 2 输入与门和两个反相器构成，可采用 CMOS 逻辑器件实现。其中的 2 输入与门可用一个 2 输入与非门加一个反相器来代替。

电路中 EN 端是振荡器的控制输入端，当 EN 为低电平时，与门 U_1 输出恒定低电平，导致反相器 U_3 的输出端 OUT 固定输出低电平，振荡器处于停止工作状态；当 EN 为高电平时，与门 U_1 可作为一个缓冲器，其输入端 a 点的电压受到 R、C 元件上的电压控制，作用是将 a 点的模拟信号转变为 b 点的数字信号。

若 EN 恒为高电平，某一时刻中 b 点为低电平，c 点为高电平，电容 C 处于充电状态，当电容 C 充电到电压 V_{DD} 的一半时，a 点进入由低电平向高电平转换点。这时 b 点变成高电平，c 点变成低电平，OUT 端输出高电平，电路状态发生翻转。这时电容 C 开始向 c 点放电，使得 a 点电压降到电源电压 V_{DD} 约一半时，即当 a 点电压低于 V_{DD} 的一半，与门识别为低电平时，b 点转低电平，c 点转为高电平，电路状态回复到原来状态，这样周而复始地进行，OUT 输出方波脉冲。图 11-3 即图 11-2 电路的工作波形图。

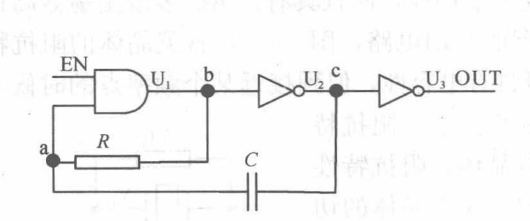


图 11-2 带使能端的非对称多谐振荡器

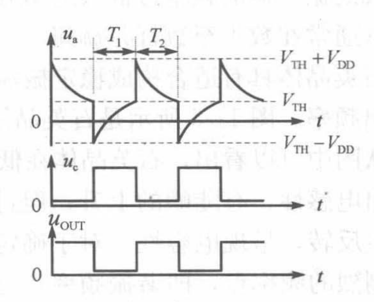


图 11-3 各点电压波形图

图 11-3 中最上面的是 a 点电压，中间为 c 点电压，最下面为 OUT 端电压。若 U_1 是 CMOS 工艺的与门，那么 a 点输入到与门的电流可以忽略不计，而且图 11-3 中的 $V_{TH}=1/2\ V_{DD}$，那么 RC 一次充/放电时间 T_{RC}（T_1 或 T_2）的计算可以依据以下公式：

$$T_{RC} \approx RC\ln\frac{V_{DD}-(V_{TH}-V_{DD})}{V_{DD}-V_{TH}} = RC\ln 3 \approx 1.1RC$$

OUT 端输出一个脉冲的周期事实上是由两个 T_{RC} 构成，即 OUT 输出脉冲的周期为

$$T_{OUT} \approx T_1 + T_2 \approx 2 \times 1.1RC = 2.2RC$$

于是，可由信号周期获得此电路的脉冲频率为

$$f_{OUT} = 1/T_{OUT} \approx 1/(2.2RC)$$

如果不需要对脉冲电路有使能控制，可以将与门换成一个反相器，其输出脉冲的频率计算与图 11-2 电路的计算方法相同。

11.1.3 对称型多谐振荡器

相比于电路图 11-2，图 11-4 是一个对称型多谐振荡器，这主要是电路结构上的对称性。对称型多谐振荡器的工作原理与非对称式多谐振荡器类似，也是通过 RC 电路的充放电使电路在两个暂稳态间进行交替变换，从而输出矩形脉冲。

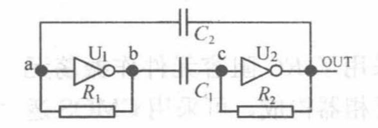


图 11-4 对称多谐振荡器

图 11-4 中的反相器可以使用 TTL 器件，也可使用 CMOS 器件。此电路结构是一个经典的多谐振荡器，其输出脉冲周期的计算涉及具体电路电压传输特性分析，分析过程可参考相关资料。

11.1.4 石英晶体振荡电路

实际上，在绝大多数的数字系统中，普遍采用石英晶体振荡器作为数字系统时钟源，而不是以上讨论的多谐振荡器。这是因为，用 RC 元件和门电路构成的多谐振荡器的输出脉冲频率受电容、电阻、门延迟值的离散性的制约，难以做到非常精确；特别是当电源电压、环境温度、气压等因素发生波动时，将严重影响时钟频率的稳定性。除了对时钟频率的稳定性有特别高的要求外，现代数字系统对工作时钟的频率也有较高的要求。由 RC 电路构成的振荡器的频率过低（通常在数万赫），没有实用意义，远远无法适应数字系统的要求（通常在数十至数百兆赫间）。

石英晶体具有适合构成稳定振荡的阻抗频率特性，而且具有比 RC 多谐振荡器高得多的输出频率。图 11-5 所示是石英晶体振荡器的常用电路，图 11-6 是石英晶体的阻抗特性图。从图中可以看出，石英晶体在低频率段具有电容性，但到接近某个频率点的时候，又体现出电感性，有陡峭的上升；但过了此频率点后，阻抗特性发生反转，呈现电容性。对于确定的石英晶体，阻抗特性变化剧烈的频率点，即谐振频率点是固定的，仅与晶体的切割方向、机械尺寸相关，具有非常高的精确度与稳定度。这个稳定度的数量级可以达到 10^{-10} 以上，完全可以满足绝大部分数字系统的时钟要求。

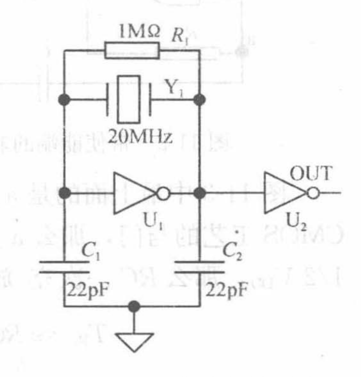


图 11-5 石英晶体振荡器电路

图 11-5 所示的石英晶振的谐振频率是 20MHz，反相器 U_2 具有波形整形的功能，其输出端的方波脉冲频率为 20MHz。

如果数字系统中需要多个频率的时钟源，可以采用振荡

器加分频器的集成芯片，如 4060、74HC4060 等，其典型电路如图 11-7 所示。这些器件的内部带有可外接石英晶体振荡器的电路，其结构与图 11-5 相同。电路的主要元件是一个反相器 U_1 和并联电阻 R_1，然后加一个 14 级的分频电路。该电路可将石英晶体振荡的频率作 $2^4 \sim 2^{14}$ 的分频。图 11-7 中的 $Q_4 \sim Q_{14}$ 是分频信号输出，例如，其中的 Q_4 对应于第 4 级分频输出，即输出频率值是输入频率值的十六分之一。

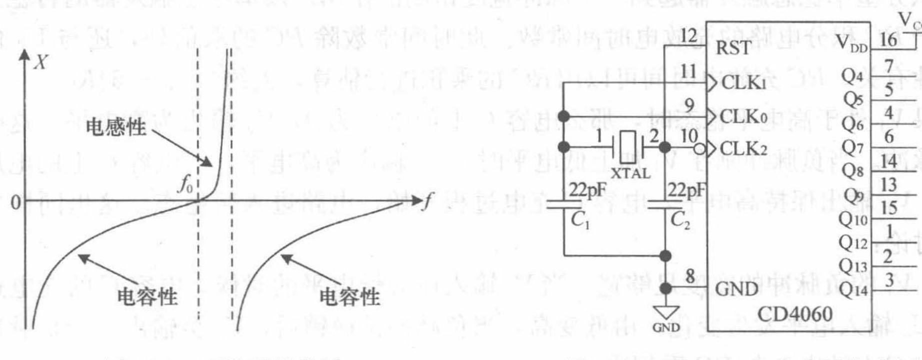


图 11-6　石英晶体特性　　图 11-7　CD4060 芯片应用电路图

11.2　单稳态触发器

多谐振荡器的工作方式是在两个暂稳态上反复切换，从而构成振荡。而以下介绍的单稳态触发器的工作状态则是由一个稳态和一个暂稳态构成。在稳态时，可以在外加脉冲的触发激励下，转到暂稳态，再经过一段时间间隔，自动恢复到稳态。其暂稳态的维持时间取决于电路参数。在早期的数字系统中，单稳态触发器常用于脉冲信号整形、脉冲定时、脉冲信号延迟等。但是，由于单稳态触发器不是一个纯粹的数字电路，而是一种模数混合电路，其稳定性、精确性以及体积方面都不再适应现代数字系统了。

11.2.1　积分型单稳态触发器

图 11-8 显示的单稳态触发器电路属于典型的积分型单稳态电路，它由一个反相器、一个与非门及 RC 积分电路构成。当 V_i 为低电平时，U_1 的输出端为高电平，V_o 输出为高电平，电容 C 开始充电。电容达到稳态时充电到高电平电压。这时，如果 V_i 输入一个正脉冲，即 V_i 由低变高，U_1 的输出端变成低电平，但由于电容 C 上面的电压不能突变，依然是高电平电压，V_o 输出电压变为低电平，电路进入暂稳态，这时电容 C 通过电阻 R 放电。这里，根据输入正脉冲的宽度，可以分两种情况讨论：

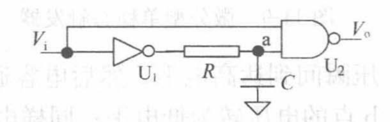


图 11-8　积分型单稳态触发器

(1) V_i 正脉冲的宽度足够宽。当电容 C 放电到能够使得 U_2 发生反转的电压值时，输出 V_o 变为高电平，暂稳态结束。事实上，这时如果正脉冲的高电平维持的时间再延长一些，也不会改变稳态的结果了，即该单稳态触发器输出的低电平的宽度是固定的，这时及

以后的时间里它只取决于该电路的参数。

(2) V_i 正脉冲的宽度不够宽。当电容 C 还未放电结束，V_i 的高电平已经结束，V_o 将立即变为高电平，电容 C 再次进入充电状态，直到充电结束，电路再次回到稳态。对于这样的窄脉冲，该单稳态电路就如同一个缓冲器。

综合上述两种情况，可以发现，从电路处于 V_i 为低电平时的稳态时，输入一个正脉冲，该积分型单稳态触发器起到一个脉冲宽度钳制的作用。该单稳态触发器的暂稳态的时间取决于 RC 积分电路的充放电时间常数。此时间常数除 RC 的取值外，还与 U_2 的电压传输特性有关。RC 充放电时间可以用 RC 的乘积进行估算，大约为 $(3\sim5)RC$。

如果 V_i 处于高电平稳态时，那么电容 C 上的电压为 0，V_o 输出为高电平，这时输入一个负脉冲。当负脉冲刚在 V_i 加上低电平时，U_1 输出为高电平，但电容 C 上的电压仍为低电平，V_o 输出保持高电平，电容 C 充电过程开始，电路进入暂稳态。这里同样要分两种情况讨论：

(1) V_i 的负脉冲的宽度足够宽。当 V_i 输入还为低电平的时候，电容 C 的充电过程已经使得 U_2 输入电平发生变化，由低变高，当负脉冲被撤销后，V_o 会输出一个负脉冲，宽度由电容 C 上的电压与 RC 乘积决定。

(2) V_i 的负脉冲的宽度比较窄。电容 C 的充电过程还没有结束，V_i 就回复到高电平，那么输出 V_o 会一直保持高电平，该输入的负脉冲被单稳态电路吸收，这个特点可以用来滤除脉冲中的毛刺。

在现代数字系统中，滤除毛刺不会再采用上述积分型单稳态电路了，而是采用同步时序电路来解决电路输出脉冲的毛刺。同样，控制脉冲宽度也未必使用单稳态这样的慢速电路。

11.2.2 微分型单稳态触发器

图 11-9 给出了一种微分型单稳态触发器的电路结构，它由一个反相器、一个或非门和两个构成微分电路的 RC 元件构成。当电路处于稳态时，电阻 R_1、R_2 都没有电流流过，故 a 点为低电平，c 点为高电平；V_o 输出为低电平，导致 b 点为高电平。

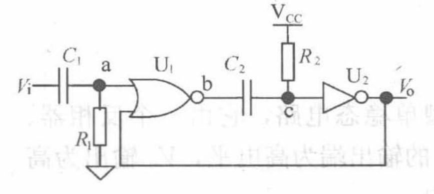


图 11-9 微分型单稳态触发器

当 V_i 输入一个正脉冲，V_i 端的电平由低电平转为高电平；由于电容 C_1 上的电压不能突变，a 点电压瞬间到达高电平，然后电容通过电阻 R_1 进行充电。由于 U_1 的一个输入端变为高电平，b 点的电压转为低电平；同样电容 C_2 上的电压不能突变，c 点的电压瞬间到达低电平，电容 C_2 通过电阻 R_2 开始充电；V_o 端输出高电平，反馈回 U_1，维持 U_1 的低电平输出。当 C_1 充电完毕，a 点电压回复到低电平，但 U_1 的输出端 b 点由于 V_o 高电平的反馈，继续维持低电平。当电容 C_2 充电完毕，c 点恢复高电平，V_o 输出为低电平，反馈回来使 U_1 的输出 b 点为高电平，c 点的电压被抬高，高于 V_{CC} 电压，电容 C_2 通过 R_2 开始放电，但 V_o 输出一直保持低电平，直至放电结束回到稳态。

当 V_i 输入正脉冲结束，a 点电压下降至比 GND 还低，电容 C_1 通过电阻 R_1 放电，直至放电结束，但 a 点电压一直为 U_1 的输入端，是一个低电平，不改变输出电平。

详细波形情况见图 11-10。对于图 11-9 这个微分单稳态触发器，其输入正脉冲只是一个触发脉冲，正脉冲的宽度与最后 V_o 输出的脉冲宽度无关，在 V_i 输入正脉冲时，R_1、C_1 构成的微分电路已经对脉冲做了微分，取出了脉冲正边沿。V_o 的输出与 R_2、C_2 的参数有关。R_2 和 C_2 决定了 V_o 的脉冲宽度，该脉宽与 R_1C_1 的乘积成正比。

微分型单稳态触发器的这些特性可用作脉冲信号的整形和脉冲宽度的调整，或用作定时脉冲的生成。

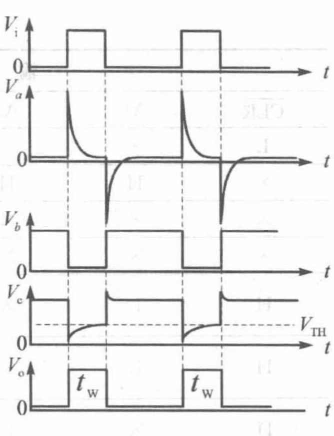


图 11-10　单稳态触发器波形图

11.2.3 集成单稳态触发器

集成单稳态触发器有非可重触发单稳态触发器和可重触发单稳态触发器两种，它们的区别在于暂稳态是否可以在输入后续脉冲的触发下，重新开始。电路在输出的表现上可见图 11-11，图中的波形（a）是一个输入触发脉冲序列，其第二个触发脉冲是在第一个触发脉冲引发的暂稳态没有结束前出现的。波形（b）是非可重触发单稳态触发器的输出。波形（c）为可重触发单稳态触发器的输出。可以看到波形（c）在第二个触发脉冲的触发下重新开始了周期为 T_w 的脉冲的输出。

集成非可重触发单稳态触发器有 74121、74LS221 等；集成可重触发单稳态触发器有 74LS122、74LS123 等，这些芯片都是 TTL 的。当然也有 CMOS 工艺的单稳态触发器。这些单稳态触发器在现代数字系统中的应用极少，其原因已在前面介绍过了。

以下主要以 74LS122 为例介绍集成单稳态触发器的使用方法。图 11-12 是 74LS122 的一种电路接法，表 11-1 是对应的功能表。74LS122 内集成了一个电阻，因此可以直接使用内部电阻作为单稳态定时元件，使用内部电阻的接法可参考图 11-13。

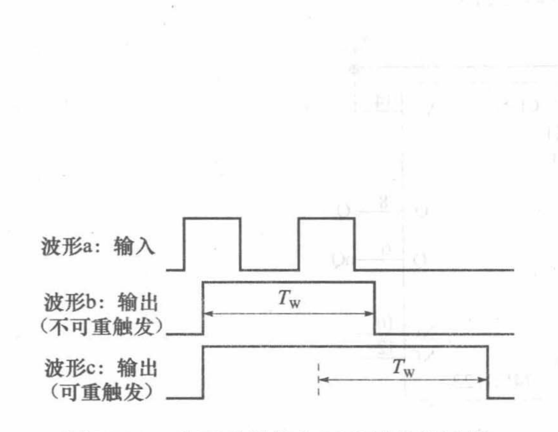


图 11-11　非可重触发与可重触发的区别

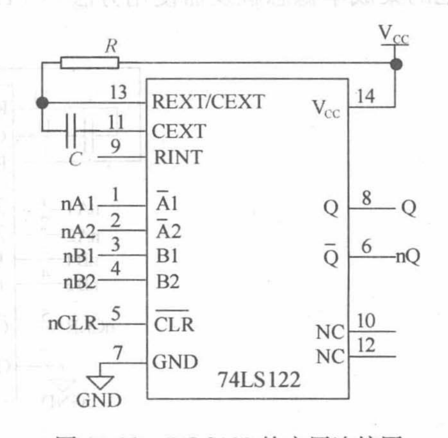


图 11-12　74LS122 的应用连接图

表 11-1 74LS122 功能真值表

输入					输出		功能
$\overline{CLR}$	A1	A2	B1	B2	Q	$\overline{Q}$	
L	×	×	×	×	L	H	复位
×	H	H	×	×	L	H	
×	×	×	L	×	L	H	
×	×	×	×	L	L	H	
H	L	×	↑	H	⊓	⊔	上升沿触发
H	L	×	H	↑	⊓	⊔	
H	×	L	↑	H	⊓	⊔	
H	×	L	H	↑	⊓	⊔	
H	H	↓	H	H	⊓	⊔	下降沿触发
H	↓	↓	H	H	⊓	⊔	
H	↓	H	H	H	⊓	⊔	
↑	L	×	H	H	⊓	⊔	上升沿除非
↑	×	L	H	H	⊓	⊔	

如果按照图 11-12 电路连接，74LS122 的 Q 端输出的脉冲宽度 T_w 可按下式计算：

$$T_w = K \cdot R \cdot C \cdot (1 + 0.7/R)$$

其中 $K=0.32$，R 为外接电阻，C 为外接电容。如果采用图 11-13 电路连接，则 R 用内部电阻的阻值取代，即 $R=R_{INT}$，具体阻值可参考对应器件的技术资料。

在表 11-1 中的"↑"表示上升沿跳变，"↓"表示下降沿跳变，"×"表示任意状态。其他的集成单稳态触发器使用方法与 74LS122 类似。

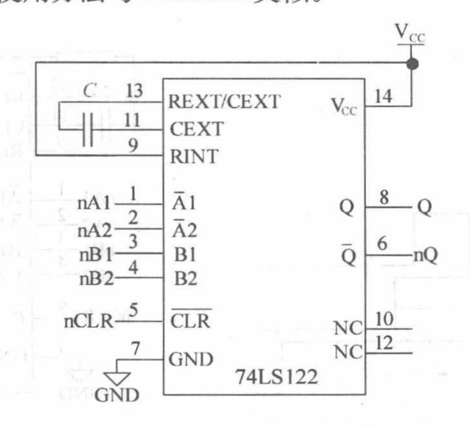


图 11-13 74LS122 的另一种应用连接图

11.3 施密特触发器

施密特触发器（Schmitt Trigger）称之为迟滞比较器，有着独特的双门限输入电平检测，常用于数字系统中的脉冲整形、干扰信号去除等。

11.3.1 施密特触发器概述

图 11-14（a）所示的是施密特触发器的图形符号，但很多情况下，施密特触发器的图形符号表示为图 11-14（b），它带有反相器的功能，也被称为施密特反相器。

下面以常见的施密特反相器的电压传输特性来分析其工作特性。图 11-15 是一个施密特反相器的工作波形图。波形显示，施密特反相器具有两个输入门限电平，当输出 V_o 为高电平时，输入 V_i 到达 V_{T+} 后，输出 V_o 才变为低电平；当 V_o 为低电平时，输入 V_i 低于 V_{T-} 后，输出 V_o 才变为高电平。在这里存在两个输入比较门限 V_{T+} 和 V_{T-}。图 11-16 更直观地表示了施密特反相器的电压传输特性。

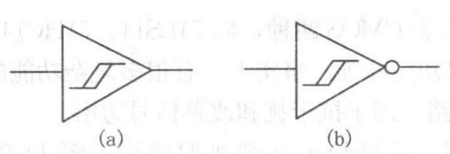


图 11-14 施密特触发器的表示

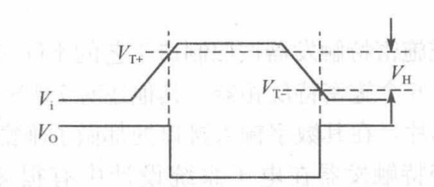


图 11-15 施密特反相器的工作波形

V_{T+} 大于 V_{T-}，且高于脉冲幅度的一半；V_{T-} 则小于脉冲幅度的一半。显然若输入波形是一个常规的矩形波形上叠加了一个幅值小于（$V_{T+}-V_{T-}$）的干扰信号时，施密特反相器可以滤除这个干扰信号。

此外，从图 11-15 的波形参数上看，若输入 V_i 有较长的上升时间和下降时间时，输出 V_o 的波形却是陡峭的，即上升时间和下降时间很短，这也是波形整形的作用。

可以使用普通的门电路来构成施密特触发器电路，如图 11-17 所示。图中显示，使用了两个反相器加两个电阻构成了施密特反相器电路，其中 $R_1<R_2$。

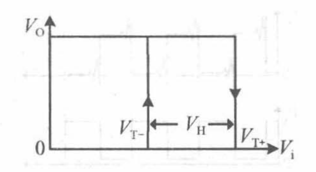


图 11-16 施密特反相器的电压传输特性

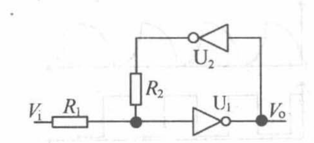


图 11-17 用反相器构成施密特电路

图 11-17 中的反相器是 CMOS 器件，设其翻转门限电平为电源电压 V_{CC} 的一半。当 V_o 为低电平时，U_2 的输出端上的电平为高电平，电阻 R_1、R_2 构成分压电路，要使反相器 U_1 获得翻转必须满足：$V_i+\frac{R_1}{R_1+R_2}\cdot(V_{CC}-V_i)<\frac{1}{2}V_{CC}$，即 $V_i<\frac{1}{2}\left(1-\frac{R_1}{R_2}\right)\cdot V_{CC}$。

当 $R_1<R_2$ 时，$V_i<\frac{1}{2}V_{CC}$才能翻转，可知：$V_{T-}=\frac{1}{2}\left(1-\frac{R_1}{R_2}\right)\cdot V_{CC}$。

一旦 $V_i<V_{T-}$，V_o 即变为高电平，U_2 的输出变为低电平，这时需要满足：$\frac{R_2}{R_1+R_2}\cdot V_i>\frac{1}{2}V_{CC}$，即$V_i>\frac{1}{2}\left(1+\frac{R_1}{R_2}\right)\cdot V_{CC}$时，输出 V_o 回复到低电平。

这个施密特电路的 $V_{T+}=\frac{1}{2}\left(1+\frac{R_1}{R_2}\right)\cdot V_{CC}$。

若使用 CMOS 工艺，施密特触发器还可按照图 11-18 来构建。这种电路采用改变 PMOS 或者 NMOS 的源极电压的方式实现不同的输入门限。这种结构不需要电阻，在集成电路中可以使用很小的面积来构建施密特触发器，使用较广泛。但该电路的 V_{T+} 和 V_{T-} 与 PMOS、NMOS 的开启电压、电源电压相关，一般是固定的。

图 11-18 CMOS 施密特触发器电路结构

11.3.2 集成施密特触发器及其应用

集成施密特触发器按照制造工艺的不同有 TTL 和 CMOS 两种，如 74LS14、74HC14 等，内部含有 6 个施密特反相器；其他还有 74LS13、CD40106 等。事实上，有很多复杂功能的数字逻辑芯片，在其数字输入接口处都做了施密特电路，用于抗干扰和改善信号边沿。

施密特触发器在电子系统设计中有很多应用，这里给出几则典型应用：图 11-20～图 11-22 分别是图 11-19 的施密特反相器电路的对应不同输入信号的输出响应波形图。

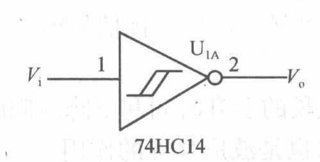


图 11-19 施密特反相器典型应用

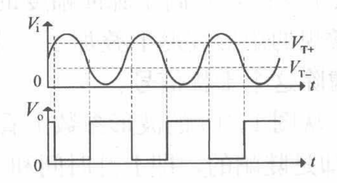


图 11-20 使用施密特反相器对正弦波整形

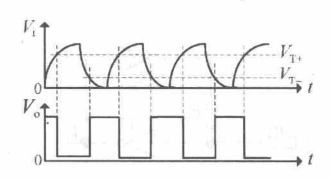


图 11-21 对不规则波形整形

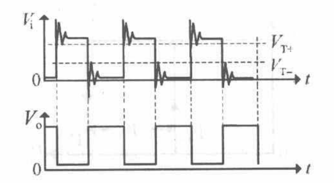


图 11-22 脉冲波上的尖峰去除

图 11-20 显示了施密特反相器电路将输入的正弦波信号整形成为 TTL 电平的矩形波；图 11-21 则显示了此电路对不规则脉冲波形的整形功能；而图 11-22 是使用施密特触发器对时钟信号中的尖峰脉冲进行了去除。

11.3.3 用施密特触发器构成多谐振荡器

还可以利用施密特反相器的双门限的特性构成多谐振荡器。若图 11-23 的电路使用了 74HC14 中的一个施密特反相器，并外接了 RC 元件，其输出频率计算公式如下：

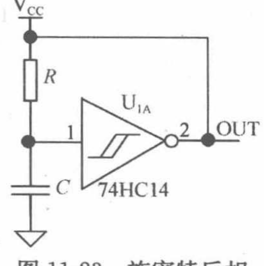


图 11-23 施密特反相器构成多谐振荡器

$$f=\frac{1}{T}\approx\frac{1}{0.8RC}$$

其中，T 为振荡周期，等于 RC 充放电的时间常数。

由于工艺参数的离散性，不同工艺的施密特反相器的输出频率计算并不相同，如果换成 74HCT14，则计算公式为

$$f=\frac{1}{T}\approx\frac{1}{0.67RC}$$

11.4 555 定时器

555 定时器是多功能的小规模数字模拟混合电路，在传统的电子系统中，常能见到 555 定时器的应用。由于其极大的灵活性，555 可以构成功能多样的实用电路。但在现代电子系统中，特别是单片机的普及及其售价的大幅降低，555 定时器早已失去了原有在电路设计中的地位，适用面越来越小。但从学习电路分析的角度看，仍有存在的价值。

555 定时器常见的型号有 TTL 工艺的 NE555、CMOS 工艺的 ICM7555、带双 555 定时器的 NE556 等。

11.4.1 555 的内部结构和工作原理

1. 555 的内部结构

555 定时器芯片的典型封装是 DIP8（8 脚双列直插封装），引脚图见图 11-24 所示。

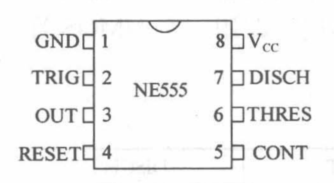


图 11-24 NE555 引脚图

图 11-25 给出了 TTL 工艺的 555 定时器内部结构原理图。图中显示，555 由两个模拟比较器、一个可控的 RS 触发器、一个集电极开路的三极管和三个阻值一致的内部电阻构成。其中两个比较器各有一个输入端分别与内部电阻分压网络相接，其输出端与 RS 触发器的 R、S 端相连。RS 触发器的一端输出通过一个与门与一个三极管的基极相接。

555 定时器 8 个引脚的功能如下：

- GND、V_{CC}：分别是电源地和电源电压输入端。
- OUT：定时器输出端。
- RESET：内部 RS 触发器复位，低电平有效。
- CONT：控制电压输入端。
- THRES、TRIG：分别进入比较器 A 和 B，是阈值电压输入端和触发输入端。

- DISCH：放电端。

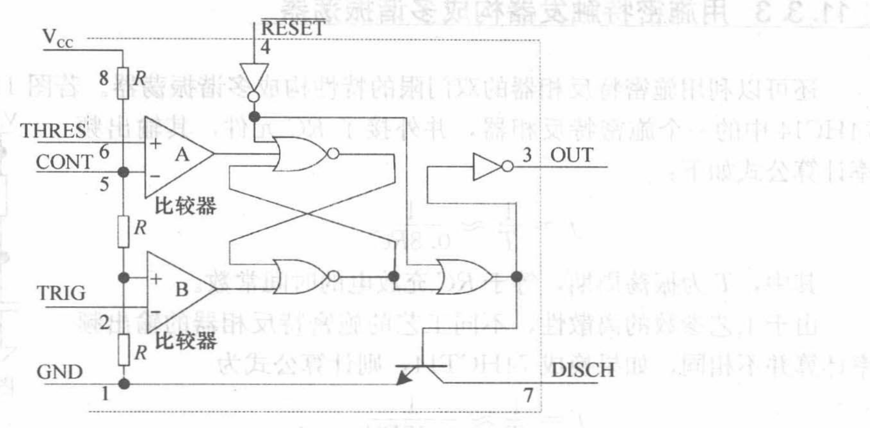


图 11-25 NE555 内部结构示意图

2. 555 的工作原理

555 的工作原理是这样的，当加上电源电压 V_{CC} 后，三个阻值一致的电阻构成分压网络，比较器 A 负端输入的电压为 2/3 V_{CC}；比较器 B 正端输入的电压为 1/3 V_{CC}。

当 TRIG 输入端输入触发电压低于 1/3V_{CC} 时，比较器 B 输出高电平，控制 RS 触发器通过 OUT 端输出高电平。可以认为比较器 B 的输出接到了 RS 触发器的 S 端，于是 THRES 输入端输入大于 2/3 V_{CC} 的电压时，比较器 A 输出高电平，该高电平控制 RS 触发器在 OUT 端输出低电平，也就是说，比较器 A 控制 RS 触发器的 R 端。

当 OUT 端输出低电平时，DISCH 端三极管导通；当 OUT 端输出高电平时，DISCH 端三极管截止（关断）。当 RESET 为低电平，RS 触发器被直接清 0，OUT 输出端输出低电平；当 RESET 为高电平，TRIG 上的电压大于 1/3 V_{CC}，而且 THRES 上的电压小于 2/3 V_{CC} 时，RS 触发器内部状态保持不变，输出端维持原来状态。

表 11-2 给出了 555 的功能表。观察表 11-2 可以发现，TRIG 端的控制优先级要高于 THRES，当 TRIG 端电压小于 1/3 V_{CC} 的时候，无论 THRES 是什么状态，OUT 输出端电压恒定为高电平。对于 CMOS 工艺的 555，其内部结构稍有不同，即用一个 NMOS 管代替了图 11-25 下方的三极管。

表 11-2 NE555 功能表

RESET 复位	TRIG 触发电压	THRES 阈值电压	OUT	DISCH
L	×	×	L	导通
H	$<1/3\ V_{CC}$	×	H	关断
H	$>1/3\ V_{CC}$	$>2/3\ V_{CC}$	L	导通
H	$>1/3\ V_{CC}$	$<2/3\ V_{CC}$	保持	保持

11.4.2 用 555 构成施密特触发器

使用 555 构成施密特触发器的电路如图 11-26 所示，电路中直接把 THRES 端和

TRIG端连接到一起作为施密特触发器的输入，施密特触发器的输出端就是OUT端。

图11-27显示，V_i输入的是正弦波，在波形电压由低往高变化时，电压值只有超过2/3 V_{CC}，即THRES端的电压大于2/3 V_{CC}时，555中的RS触发器被清0，V_o输出低电平；而在正弦波幅值下降到小于2/3 V_{CC}，而大于1/3 V_{CC}时，555的输出处于维持状态，V_o仍然保持低电平；在幅值继续回落到1/3 V_{CC}以下，满足TRIG端电压小于1/3 V_{CC}时，555输出发生翻转，V_o电压变为高电平。由此分析可知，这是个$V_{T+}=2/3\ V_{CC}$，$V_{T-}=1/3\ V_{CC}$的施密特触发器。如果需要修改555构成的施密特触发器的V_{T+}和V_{T-}，可以通过CONT端进行调整，即可在CONT端上加上合适的控制电压。

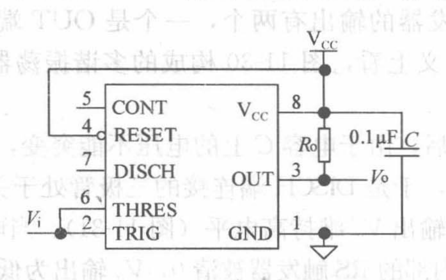


图11-26 用555构成施密特触发器

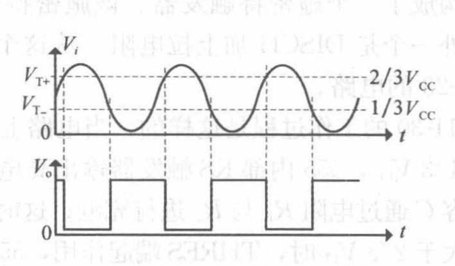


图11-27 输入输出波形图

11.4.3 用555构成单稳态触发器

555也可构成单稳态电路，其电路的连接方式如图11-28所示。电路中，TRIG用作触发脉冲输入端，THRES端接一电容到地，DISCH端反馈到THRES端并接一电阻到V_{CC}。

图11-29显示了用555构成的单稳态电路的工作波形。图11-29（c）的波形是电容C_1上的波形。当V_i输入一个触发脉冲（注意这个触发脉冲是一个负脉冲，这是因为TRIG低于1/3 V_{CC}时有效），555内部RS触发器被设置为高电平输出，这时连接到DISCH端的三极管关断，电容C_1通过电阻R_1进行充电，单稳态电路进入暂稳态。这时，当C_1上的电压上升到2/3 V_{CC}，THRES端输入信号有效时，对555内部RS触发器进行了清0，DISCH端连接的三极管导通，电容C_1上的电压被快速地释放掉，内部RS触发器一直保持低电平输出，单稳态电路回复到了稳态。

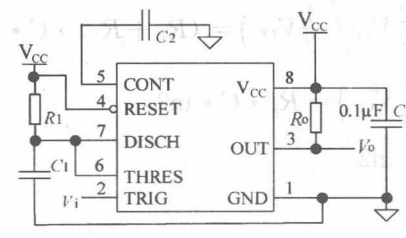


图11-28 用555构成的单稳态电路

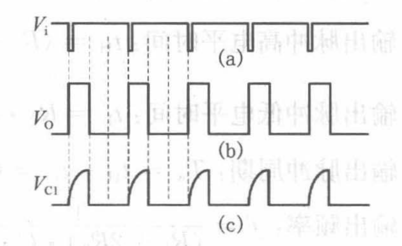


图11-29 图11-28的波形图

单稳态电路在触发后输出的脉冲宽度由RC元件从0V充电到2/3 V_{CC}的时间决定。这

个时间与 V_i 输入的触发脉冲的宽度无关，即只与下降沿的触发有关。这是个非可重触发型的单稳态触发器。

11.4.4 用555构成多谐振荡器

用555定时器还可构成多谐振荡器，常用的电路结构如图11-30所示，对应的输出波形图如图11-31所示。

电路中，THRES端与TRIG端相连，电路结构与图11-26类似，也就是说这个555电路也构成了一个施密特触发器。该施密特触发器的输出有两个，一个是OUT端即 V_o 端，另外一个是DISCH加上拉电阻，从这个意义上看，图11-30构成的多谐振荡器类似于图11-23的电路。

图11-30的工作过程是这样的，当电路上电后，由于电容 C 上的电压不能突变，TRIG端小于 $1/3\ V_{CC}$，555内部RS触发器输出高电平，于是DISCH端连接的三极管处于关断状态，电容 C 通过电阻 R_1 与 R_2 进行充电；这时，输出 V_o 维持高电平（图11-31）。当电容 C 充电到大于 $2/3\ V_{CC}$ 时，THRES端起作用，555内部的RS触发器被清0，V_o 输出为低电平，DISCH端相连接的三极管导通，电容 C 通过电阻 R_2 放电，直到小于 $1/3\ V_{CC}$，内部RS触发器再次被置位，这个过程反复进行，便出现了振荡。

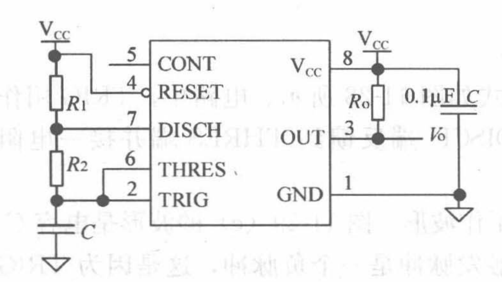


图11-30 555构成多谐振荡器

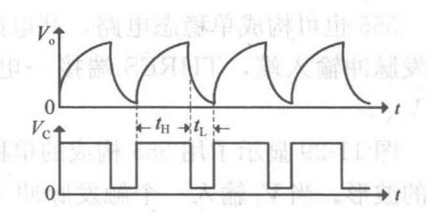


图11-31 555构成的多谐振荡器的波形图

电路在振荡状态时，电容 C 上的电压在 $1/3\ V_{CC}\sim 2/3\ V_{CC}$ 间交替变化，这期间，电容 C 的充电是通过 R_1 与 R_2 进行的，然而放电则仅通过 R_2 完成，所以该电路输出的波形的占空比不会是50%的等宽度。

电路波形参数的计算如下：

输出脉冲高电平时间：$t_H=(R_1+R_2)\cdot C\cdot \ln\left(\frac{2}{3}V_{CC}\Big/\frac{1}{3}V_{CC}\right)=(R_1+R_2)\cdot C\cdot \ln 2$

输出脉冲低电平时间：$t_L=R_2\cdot C\cdot \ln\left(\frac{2}{3}V_{CC}\Big/\frac{1}{3}V_{CC}\right)=R_2\cdot C\cdot \ln 2$

输出脉冲周期：$T_w=t_H+t_L=(R_1+2R_2)\cdot C\cdot \ln 2$

输出频率：$f=\dfrac{1}{(R_1+2R_2)\cdot C\cdot \ln 2}$

输出脉冲占空比：$q=\dfrac{t_H}{t_H+t_L}=\dfrac{R_2}{R_1+2R_2}$

习 题

11-1 试举出三种多谐振荡器的名称，并且分析其工作原理。

11-2 试构建一个石英晶体振荡器，要求输出频率为 20MHz。

11-3 根据 11.2.1 节的讨论和电路图 11-8，对应 V_i 不同的脉冲输入，分别绘出对应 V_i 的 a 点和输出点（V_o）的波形。

11-4 使用 NE555 构成单稳态触发器，触发后输出脉冲宽度为 100μs，画出电路原理图和工作波形图。

第12章

实用数字系统综合设计实践

由于历史的原因，基于传统数字电路手工设计技术的设计对象，基本属于小规模、低速、多器件组合结构的简单电子线路模块，因此，这些模块的开发中面对的问题要简单得多，实用范围当然也要小得多。例如第 7 章 7.3 节中介绍的时序电路设计方法，有一个非常程式化的设计流程，完成后的电路的检验也十分简单，甚至直接从电路图结构便可以检验设计的功能特性。这是因为设计对象的规模最多只涉及数十个逻辑门的电路，因此即使有意创新，也很难施展。

相对于小规模的数字电路，数字系统的逻辑规模最大可至数百万门。对此，传统的数字技术已失效。例如对几十万逻辑门规模构成的数字系统的功能和时序的分析，显然不能用传统的直接分析电路原理图的方式来完成，而现代实用的数字电子产品的开发都离不开基本的数字系统的开发。因此，在对基本的数字电子技术的学习和入门后，必须迅速地转向熟悉基于现代数字电子设计技术的基本方法和基本理念。不但如此，更应在学习与实践中培养对数字系统设计的自主创新能力。

本章并不准备如同传统数字电子技术那样，针对数字系统给出一个程式化的设计流程和分析流程，因为这显然会限制人的想象力，破坏人的创造力。本章只期望通过介绍数个实用的数字系统的设计思路和设计方法，给出一些对应的实验与实践要求，让读者自己去探寻掌握数字系统设计技术及其创新的途径。

12.1 6 位十进制数字频率计设计

从本设计项目的系统结构上看，整个电路都可以由传统的 74 系列器件构成，但在设计工具和流程上看，系统的层次化设计及时序仿真工具的应用，充分体现了现代数字系统设计工具的强大功能。本项目的一个创新点是，利用 74 系列器件巧妙地完成了频率计的测频时序控制信号发生器的设计。以下给出项目的设计内容、流程、验证结果和实验任务。

1. 测频原理

图 12-1 是本设计项目的频率计模型框图，图中显示，除了数码管外，其他模块都在 FPGA 内部。最上方是 6 个 7 段数码管，可以显示 6 位十进制或十六进制的测频数据（如果译码器可以对十六进制数译码）。每一个数码管下方是一个 7 段译码器（在 FPGA 内），可以对输入的 4 位二进制数进行十进制或十六进制 7 段译码。模块 REG24B 是一个 24 位

的寄存器，用于锁存测频后需要显示的数据。模块CNT6D是一个6位十进制计数器，它由一个24位二进制计数器构成，用于对被测信号的脉冲进行计数。模块FT_CTRL是测频控制模块，是频率计测频时序信号发生器。

根据频率的定义和频率测量的基本原理，测定信号的频率须有一个脉宽为1s的输入信号脉冲计数允许的信号，用于控制模块CNT6D的计数时间；在允许的1s计数结束后，计数值必须锁入锁存器，计数器即清0，为下一测频计数周期做好准备。测频控制信号可以由一个独立的发生器（时序控制信号发生器）来产生，这就是图12-1中的FT_CTRL。

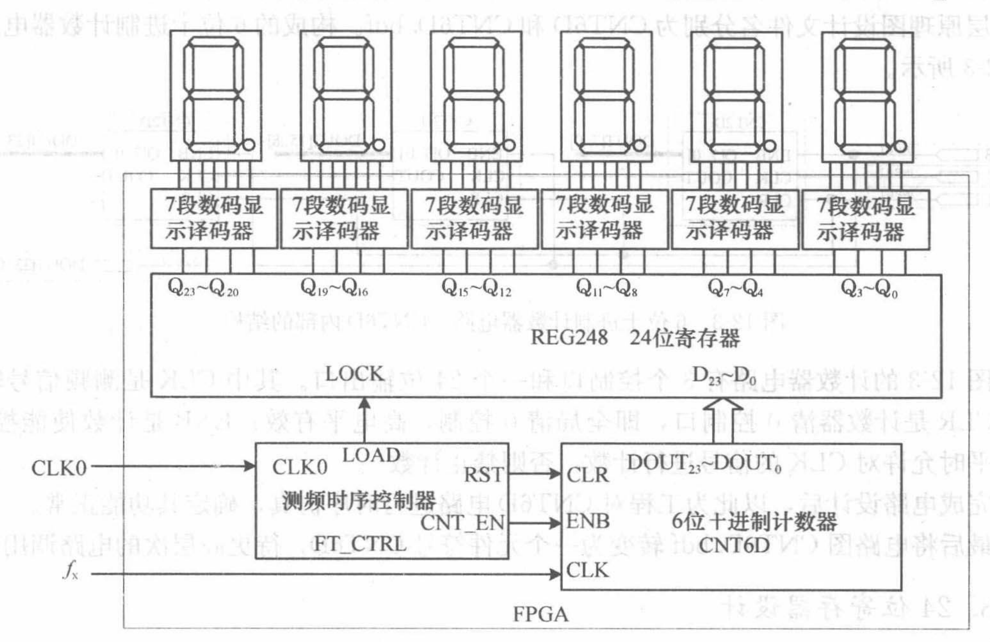


图12-1 频率计模型框图

根据测频原理，测频控制时序可以如图12-2所示，其中的f_x是被测信号，CLK0是产生测频控制时序的时钟信号。设计要求是，FT_CTRL的计数使能信号CNT_EN能产生一个1s脉宽的周期信号，并对频率计中的24位二进制计数器CNT6D的ENB使能端进行同步控制。当CNT_EN高电平时允许计数；低电平时禁止计数，并保持其所计的脉冲数。在停止计数期间，首先需要有一个锁存信号LOAD的上跳沿将计数器在前一秒钟的计数值锁存进锁存器REG24B中，并由外部的7段译码器译出，显示计数值。设置此锁

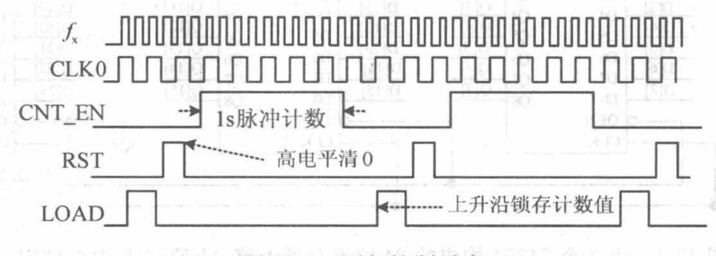


图12-2 测频控制时序

存器的目的是使数据显示稳定，不会由于周期性的清0信号而不断闪烁。锁存信号后，必须有一清0信号RST对计数器进行清0，为下一秒的计数操作做准备。

2. 6位十进制计数器的设计

可以利用8.11节中用74390设计好的2位十进制计数器（图8-1）来构成6位十进制计数器。首先将此计数器作为元件入库，元件取名为CNT2D。然后构建一个新的工程，在此工程中调用已入库的元件CNT2D，用其构成所需要的电路。设这个新的工程名和对应顶层原理图设计文件名分别为CNT6D和CNT6D. bdf。构成的6位十进制计数器电路如图12-3所示。

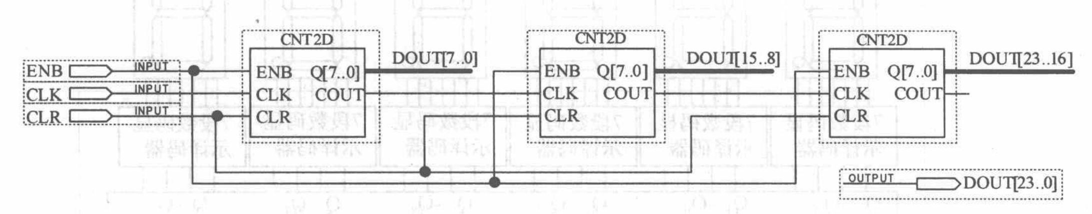


图12-3 6位十进制计数器电路（CNT6D内部的结构）

图12-3的计数器电路有3个控制口和一个24位输出口。其中CLK是测频信号输入口；CLR是计数器清0控制口，即全局清0控制，高电平有效；ENB是计数使能控制，高电平时允许对CLK的信号进行计数，否则禁止计数。

完成电路设计后，以此为工程对CNT6D电路进行时序仿真，确定其功能正常。

最后将电路图CNT6D. bdf转变为一个元件符号CNT6D，待更高层次的电路调用。

3. 24位寄存器设计

图12-4是由3个74374构成的24位寄存器电路，文件名可设为：REG24B.bdf。完成电路设计后，可以对此电路进行时序仿真。此电路的端口比较简单，一个24位数据输入口D[23..0]、一个24位数据输出口Q[23..0]，以及一个锁存控制信号LOCK，上升沿有效。同样，最后将此电路REG24B. bdf转变成元件符号REG24B，待更高层次的电路调用。也可以按照8.5节的流程直接用LPM模块来构建：首先进入如图8-32所示窗口，选择左栏的Storage项，再选其中的LPM_FF；位数设置是24。

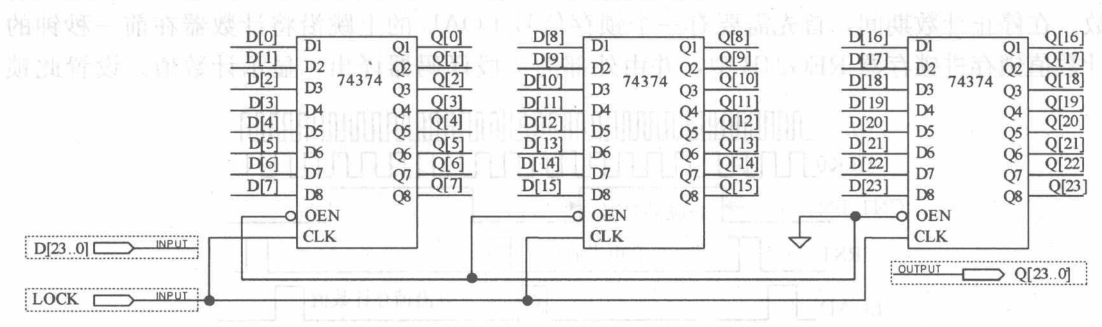


图12-4 由3个74374构成的24位寄存器电路（REG24B内部结构）

4. 时序控制器设计

对于模块 FT_CTRL 的功能要求是，能按照图 12-2 所示的时序关系，产生 3 个控制信号：CNT_EN、LOAD 和 RST，以便使频率计能分别顺利地完成计数（计数使能 CNT_EN）、锁存（计数锁存 LOAD）和清 0（计数器清 0 RST）3 个重要的功能。而且其中作为周期信号 CNT_EN 的高电平脉宽必须长 1s，以便控制图 12-3 电路的 ENB。

能产生类似图 12-2 波形的电路可以有多种，根据 3 个控制信号的时序要求，这里选用图 12-5 所示的电路。设该电路的文件名不妨取为：TF_CTRL. bdf。

可以首先建立一个新的工程，工程名取为 TF_CTRL。在此原理图编辑窗中根据图 12-5 完成电路设计。该电路由 3 部分组成：4 位二进制计数器 7493、4-16 译码器 74154 和两个由双与非门构成的 RS 触发器。只要控制好 CLK0 的频率，其中的 74154 也可以用 3-8 译码器 74138 等代替，读者不妨一试。

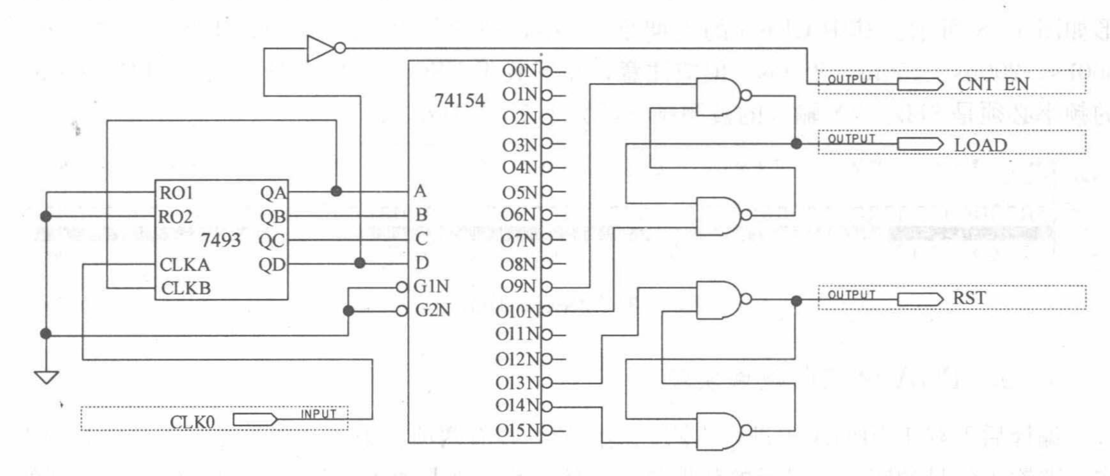


图 12-5　频率计测频时序控制器电路

根据图 12-5 的电路结构分析，如果 CLK0 的输入频率是 8Hz，则此电路的 CNT_EN 输出信号的频率为 0.5Hz，脉宽为 1s，满足设计要求。

此电路的仿真时序波形如图 12-6 所示，由此波形可以清楚地了解电路测频的工作原理。显然，通过这个时序，能自动控制类似图 12-1 的电路，实现频率测试的目的。（为什么图 12-6 所示的输出信号 RST 和 LOAD 的前一段的波形有不确定标志的网状图形?）

同样可将图 12-6 变成一个底层可调用的元件：TF_CTRL。

事实上，图 12-5 所示的电路还有许多其他用途。例如可构成高速时序脉冲发生器。可通过输入不同频率的 CLK0，或将 RS 触发器接在 74154 的不同输出端，从而产生各种不同脉宽和频率的脉冲信号。

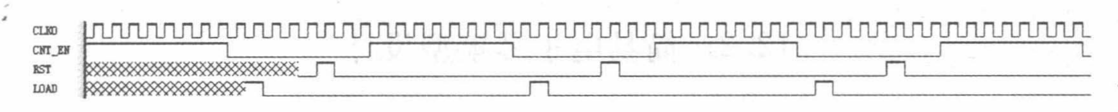


图 12-6　图 12-5 电路的仿真波形

5. 顶层电路设计与测试

最后完成整个频率计系统的顶层工程的设计。仍然使用原理图输入方式，文件名可设为：TOP. bdf。然后，在此文件的原理图编辑窗中分别调出以上已经设计好的 3 个元件：TF_CTRL、CNT6D、REG24B。然后按照图 12-7，完成频率计的最后设计。

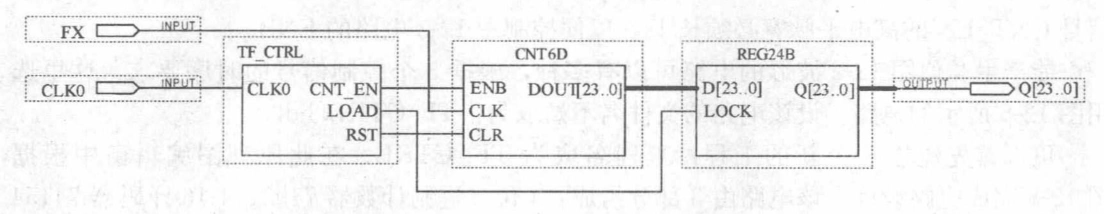


图 12-7 频率计顶层电路原理图

对于图 12-7，必须进行时序仿真，以确定其是否能正常工作。图 12-7 电路的仿真波形如图 12-8 所示。其中 CLK0 的周期是 800ns，被测信号 FX 前 5 段周期分别取 50ns、300ns、200ns、150ns、250ns。但应注意，在下载于 FPGA 中作为频率计实测时，CLK0 的频率必须是 8Hz。FX 输入的被测频率可大于 100MHz。

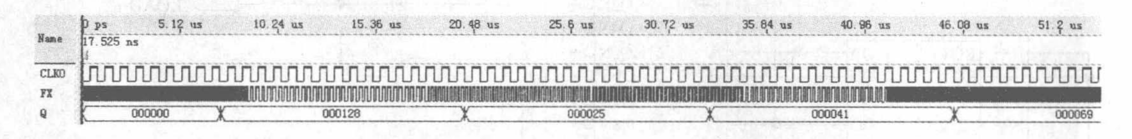


图 12-8 频率计时序波形

6. 在 FPGA 中完成硬件实测

编译后下载于 FPGA 中进行实测。实测的内容有两项：第一，能否完成正常的测频工作，即输入信号的频率与显示的数据是否一致，稳定性是否好；第二，被测信号能达到的频率上限是多少，与计算机的时序分析结果的一致性如何。实测时 CLK0 的频率必须是 8Hz。如果没有，可以从其他信号分频得到，如从内嵌的锁相环得到。

7. 设计实践任务

（1）根据以上的描述，设计一个 6 位十进制数据显示的数字频率计。测频范围是 1Hz～150MHz。给出时序仿真波形，并分析。最后进行硬件测试验证。

（2）设计 6 位十六进制数显示的数字频率计。要求图 12-7 中的 3 个模块中，用 LPM 模块实现模块 CNT6D 和 REG24B，并用其他电路方案实现模块 TF_CTRL。

编译和时序仿真，根据仿真波形说明此电路的功能，引脚锁定编译，编程下载于 FPGA 中，在实验系统上进行硬件测试。完成实验与设计报告。

12.2 简易电子琴模型设计

本设计项目的特色是知识的综合性和趣味性得到了很好的结合。本书涉及的一些数字

电路的知识多数被融入其中。

1. 系统端口

图 12-9 是电子琴顶层设计电路。电路含 2 个输入口和 3 个输出口：

(1) 工作时钟 CLK，频率：1MHz。用于在主控模块中产生与琴键对应的振荡频率，以驱动蜂鸣器发出相应的声音。如果使用了 Cyclone Ⅲ或以上系列 FPGA，此频率可通过 FPGA 内的锁相环获得。对于附录介绍的开发板，接锁相环的外部时钟频率是 20MHz。

(2) 琴键输入 DIN[7..0]。8 个音符，8 位中只能有一位为 0（低电平），即 8 个琴键中每一时刻只能按一个键。如 FIN[7..0]=11101111，则表示发出一个简谱音符“5”。

由于是演示性模型设计，只安排了 8 个音，设计者可以根据需要扩充。

(3) 输出端口 SPK 用于驱动蜂鸣器，其输出频率 f_B 与蜂鸣器所发出音调与图 12-10 所示的电子琴各音阶基频有对应关系。

(4) 输出信号 LED 接数码管，用于显示对应的简谱码。H 显示音高低。

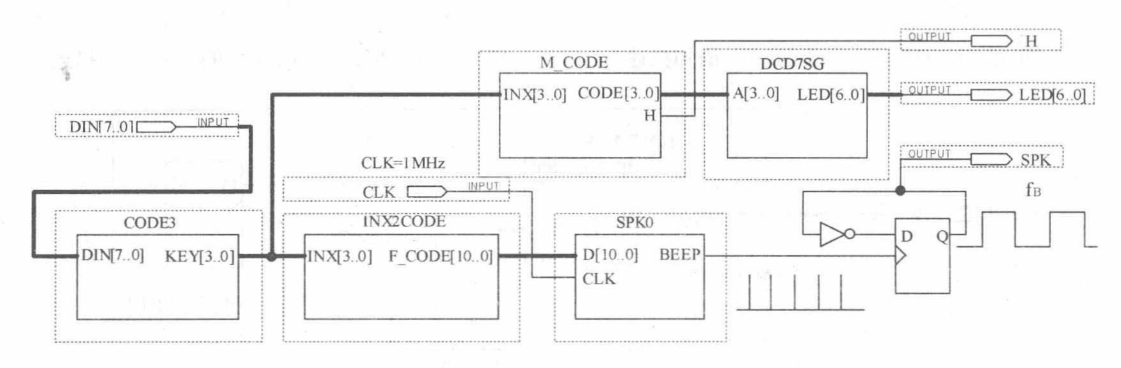


图 12-9 电子琴顶层设计电路

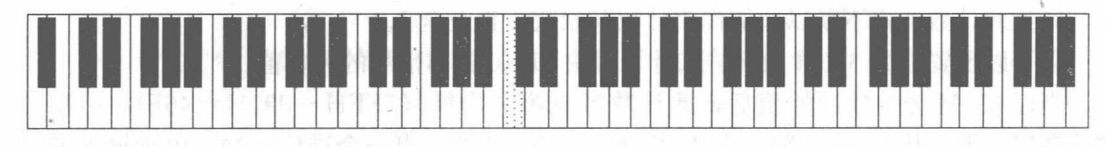


图 12-10 电子琴音阶基频对照图（单位 Hz）

2. 工作原理

图 12-9 中的模块 CODE3 是一个编码器，即将输入的 8 位琴键信号进行编码，输出一个 4 位码，最多能对应 16 个音符（若有 16 个键）。此编码器的真值表所对应的 case 语句程序如图 12-11 所示。如果增加琴键，此程序也可进一步扩充。

可以认为，模块 INX2CODE 是一个译码器，它将来自键盘输入的编码信号译码成数

控分频器 SPK0 输出信号的频率控制字。译码器模块 INX2CODE 的真值表所对应的 case 语句程序如图 12-12 所示。SPK0 的内部电路结构如图 12-13 所示。

```
module CODE3 (DIN, KEY);
  input[7:0] DIN;    output[3:0] KEY;
  reg[3:0] KEY;
  always @(DIN)
     case (DIN)
         8'b11111110 : KEY<=4'b0001;
         8'b11111101 : KEY<=4'b0010 ;
         8'b11111011 : KEY<=4'b0011;
         8'b11110111 : KEY<=4'b0100 ;
         8'b11101111 : KEY<=4'b0101;
         8'b11011111 : KEY<=4'b0110 ;
         8'b10111111 : KEY<=4'b0111;
         8'b01111111 : KEY<=4'b1000 ;
         8'b00111111 : KEY<=4'b1001;
         8'b11111111 : KEY<=4'b0000 ;
             default : KEY<=4'b0000 ;
     endcase
endmodule
```

图 12-11　CODE3 模块的 case 语句描述

```
module INX2CODE (INX, F_CODE);
  input[3:0] INX;  output[10:0] F_CODE ;
  reg[10:0] F_CODE;
  always @(INX)
  case (INX)
   0  : F_CODE <= 11'H7FF;
   1  : F_CODE <= 11'H305;
   2  : F_CODE <= 11'H390;
   3  : F_CODE <= 11'H40C;
   4  : F_CODE <= 11'H45C;
   5  : F_CODE <= 11'H4AD;
   6  : F_CODE <= 11'H50A;
   7  : F_CODE <= 11'H55C;
   8  : F_CODE <= 11'H582;
   9  : F_CODE <= 11'H5C8;
   10 : F_CODE <= 11'H606;
   11 : F_CODE <= 11'H640;
   12 : F_CODE <= 11'H656;
   13 : F_CODE <= 11'H684;
   14 : F_CODE <= 11'H69A;
   15 : F_CODE <= 11'H6C0;
  default : F_CODE <= 11'H6C0;
     endcase
endmodule
```

图 12-12　INX2CODE 模块的 case 语句描述

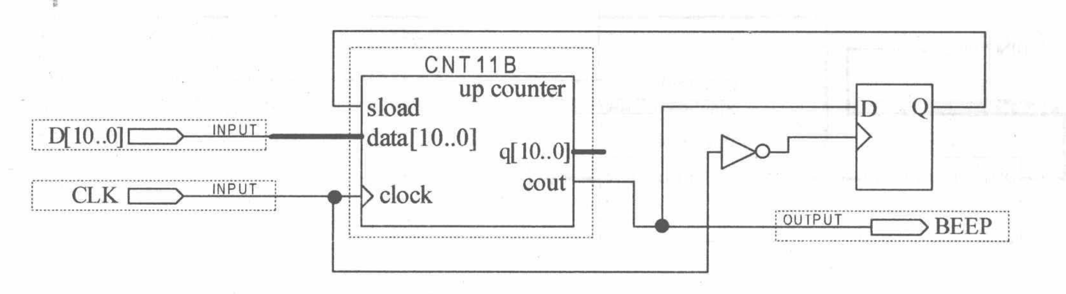


图 12-13　SPK0 模块内部电路结构

模块 CODE3、INX2CODE 和 SPK0 的主要工作过程是这样的：

当按琴键后，产生的数据经编码器获得一个编码（例如，当按下的是第二个键，对应 0010，即 2），它对应模块 INX2CODE 中的一个值（2 对应 390H）。当这个值（如 390H）被置入模块 SPK0 中的 11 位可预置计数器中后，由于计数器的进位端与预置数加载端相连，导致此计数器将不断以此值作为计数起始值，直至全 1。

以下以预置值为 390H 为例，来计算模块 SPK0 输出信号的频率值。

当以 390H 为计数起始值后，此计数器成为一个模（7FFH－390H＝46FH＝1135）的计数器。即每从 CLK 端输入 1135 个脉冲，BEEP 端输出一个进位脉冲。由于输入的时钟频率是 1MHz（周期是 1μs），于是 BEEP 输出的信号频率是 1/(11351μs)＝881Hz。

显然，INX2CODE 中的数据都可以通过这种方法，并参考图 12-10 计算出来。

从图 12-9 可见，SPK0 的输出信号经过一个由 D 触发器接成的 T′触发器后才输出给蜂鸣器。这时信号被作了二分频，于是，预置值 390H 对应的于蜂鸣器发音的基频 F_B 约等于 440Hz。这可以在图 12-10 中找到对应的音阶。

图 12-9 中的 T′触发器有两个功能，一个作用是作二分频器；另一个作用是作为占空比均衡电路。这是因为由 SPK0 模块输出信号的脉宽极窄，功率极低，无法驱动蜂鸣器。

但信号通过T′触发器后，脉宽就均匀了（F_B 的占空比为50%）。

显然，模块CODE3输出给INX2CODE的数值与驱动蜂鸣器发声的信号的频率值有直接的对应关系。

3. SPK0模块

图12-13所示的电路是模块SPK0的内部结构。其中的计数器CNT11B是一个LPM宏模块，这是一个11位二进制加法计数器。在设置其结构参数时，应该选择同步加载控制，即sload（Synchronous Load），这样能较好地避免来自进位信号cout中可能的毛刺影响。异步加载aload极易受到随机窄脉冲的误触发，在此类电路中不宜采用。图12-13中的D触发器和反相器的功能是将用于控制加载的进位信号延迟半个时钟周期，一来也是为了滤除可能的毛刺，以免对加载发生误操作，同时使加载更为可靠，因为这时，时钟上升沿正好处于加载脉冲的中点。

4. 模块M_CODE和DCD7SG

另外两个模块是M_CODE和DCD7SG，前者的功能是将来自CODE3的键盘编码译成简谱码和对应的音调高低值H，后者是一个数码管7段显示译码器，负责将简谱码译成数码管的显示信号。M_CODE和DCD7SG的程序分别如图12-14和图12-15所示。

```
module M_CODE (INX, CODE, H);
  input[3:0] INX;  output[3:0] CODE;
  output H;
  reg[3:0] CODE;   reg H;
  always @(INX)
  case (INX)
     0 : {CODE,H} <= {4'B0000,1'B0};
     1 : {CODE,H} <= {4'B0001,1'B0};
     2 : {CODE,H} <= {4'B0010,1'B0};
     3 : {CODE,H} <= {4'B0011,1'B0};
     4 : {CODE,H} <= {4'B0100,1'B0};
     5 : {CODE,H} <= {4'B0101,1'B0};
     6 : {CODE,H} <= {4'B0110,1'B0};
     7 : {CODE,H} <= {4'B0111,1'B0};
     8 : {CODE,H} <= {4'B0001,1'B1};
     9 : {CODE,H} <= {4'B0010,1'B1};
    10 : {CODE,H} <= {4'B0011,1'B1};
    11 : {CODE,H} <= {4'B0100,1'B1};
    12 : {CODE,H} <= {4'B0101,1'B1};
    13 : {CODE,H} <= {4'B0110,1'B1};
    14 : {CODE,H} <= {4'B0111,1'B1};
    15 : {CODE,H} <= {4'B0001,1'B1};
  default : {CODE,H} <= {4'B0001,1'B1};
     endcase
endmodule
```

图12-14 M_CODE模块的case语句描述

```
module DCD7SG (A,LED);
  input[3:0] A;  output[6:0] LED ;
   reg[6:0] LED  ;
  always @(A)
  case (A)
  4'B0000 :  LED <= 7'B0111111 ;
  4'B0001 :  LED <= 7'B0000110 ;
  4'B0010 :  LED <= 7'B1011011 ;
  4'B0011 :  LED <= 7'B1001111 ;
  4'B0100 :  LED <= 7'B1100110 ;
  4'B0101 :  LED <= 7'B1101101 ;
  4'B0110 :  LED <= 7'B1111101 ;
  4'B0111 :  LED <= 7'B0000111 ;
  4'B1000 :  LED <= 7'B1111111 ;
  4'B1001 :  LED <= 7'B1101111 ;
  4'B1010 :  LED <= 7'B1110111 ;
  4'B1011 :  LED <= 7'B1111100 ;
  4'B1100 :  LED <= 7'B0111001 ;
  4'B1101 :  LED <= 7'B1011110 ;
  4'B1110 :  LED <= 7'B1111001 ;
  4'B1111 :  LED <= 7'B1110001 ;
    default : LED <= 7'B1110001 ;
     endcase
endmodule
```

图12-15 DCD7SG模块的case语句描述

5. LPM_ROM型音符预置数存储器设置

为了节省逻辑资源，完全可以用一个LPM_ROM来替代模块INX2CODE。这首先必须参考图12-12，编辑一个mif格式的参数文件。文件结构如图12-16所示。

然后编辑一个LPM_ROM。此ROM的数据宽度选择11位，数据深度选择32位（最低可选择数），对应5位地址。连线时可设最高位接0。

```
WIDTH = 11 ;
DEPTH = 32 ;
ADDRESS_RADIX = DEC ;
DATA_RADIX = HEX ;
CONTENT  BEGIN
00: 7FF ; 01: 305 ; 02: 390 ; 03: 40C ; 04: 45C ; 05: 4AD ; 06: 50A ; 07: 55C ; 08: 58C ;
09: 5C8 ; 10: 606 ; 11: 630 ; 12: 656 ; 13: 684 ; 14: 69A ; 15: 6C0 ; 16: 6D6 ; 17: 6EA ;
18: 6FE ; 19: 717 ; 20: 726 ; 21: 78A ; 22: 000 ; 23: 000 ; 24: 000 ; 25: 000 ; 26: 000 ;
27: 000 ; 28: 000 ; 29: 000 ; 30: 000 ; 31: 000 ;
END ;
```

图 12-16 音符预置数 mif 配置文件

6. 时序仿真测试与硬件实现

在设计过程中，对于每一个功能模块的设计都必须进行时序仿真，以便及时了解设计的情况。如果仿真都能通过，硬件测试就成了不可或缺的工作了，这尤其是对于电子琴设计项目的功能测试，仅凭仿真是远远不够的，重要的是能听到乐声。

有关的引脚锁定方法和下载测试方法这里就不再重复了。

7. 设计实践任务

(1) 根据以上的讨论完成电子琴设计，对设计进行时序仿真，根据仿真波形分析说明各模块电路特性，编程下载于 FPGA 中，在实验系统上进行硬件测试，完成实验报告。

(2) 对本节设计作一些扩充，包括增加键的数量和对应的译码器中的数值，以及为方便地获取系统的工作时钟，增加一个内置锁相环，确保可以弹奏一些简单乐曲。

(3) 给出一个简洁的方案，根据图 12-10，方便地获取图 12-16 中的数据，并在电子琴上验证这些数据。

(4) 为图 12-9 所示的电路增加一两个 LPM_RAM，用于在乐曲弹奏时同步记录所弹乐曲的音符和节拍（提示：还要为此配置一个计数器，作为记录节拍的定时器）。最后，可通过开关控制，使电路能自动复述演奏被弹奏过的乐曲。

(5) 为扩充琴键的数量，以及琴键信号进入电子琴电路的方式，查阅相关资料，设计一些辅助电路实现之。例如为用作琴键的 4×4 矩阵键盘，增加一个键信号识别模块，以及对应的编码器；或为了将 PS2 接口的普通计算机键盘用作琴键，设计一个 PS2 通信模块，以及对应的编码器；等等。

(6) 为增加趣味性，查阅相关资料，为此电子琴模型增加一个 VGA 显示控制模块，使得在弹奏时，VGA 显示器能同步显示琴键的变动，甚至五线谱音符的跳动，等等。

12.3 乐曲自动演奏电路设计

有了模型电子琴电路的设计经历，完成乐曲的自动演奏就比较简单了。因为电子琴的演奏是通过人来弹琴完成的，而自动演奏则用电路模块代替人来完成演奏任务。

1. 电路结构与原理

如果由人来利用图 12-9 所示电路完成一首乐曲的弹奏需要对两个方面进行控制，即

音素（音符）和节奏，或节拍。因此，完成自动演奏必须增加两个功能模块，一个根据曲目输出音素，另一个控制每一音素发音的长短，即节拍。换言之，组成乐曲的每个音符的发音频率值及其持续的时间是乐曲能连续演奏所需的两个基本要素，即音准与节拍。

与图 12-9 相比，图 12-17 所示的电路增加了两个模块，即 CNTSTEP 和 MUSIC_DATA。

音符的持续时间需要根据乐曲的速度及每个音符的节拍数来确定，如前所述，模块 INX2CODE 等效于乐曲简谱码对应的分频预置数查表电路，其中设置了所有乐曲可能出现的全部音符所对应的分频预置数。

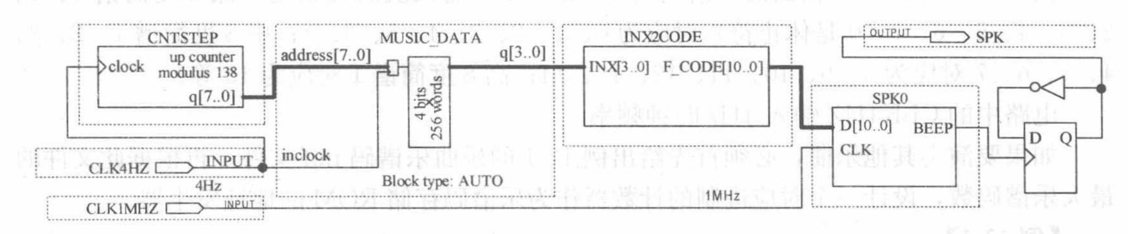


图 12-17 乐曲自动演奏电路

模块 MUSIC_DATA 是乐谱码存储 ROM。与 INX2CODE 不同，这个乐谱 ROM 是放整个乐曲乐谱码的，而且一个节拍的简谱码占据一个地址。这是因为，如果需要发“3”的音，这个音长占 4 个节拍，CLK4HZ＝4Hz，则每个节拍占时间为 0.25s，全音符发完需时 1s，为四四拍的 4 分音符持续时间。这样一来，此“3”的码必须占用乐谱码 ROM 共 4 个单元。因此，如果将一首乐曲按每个节拍（占 0.25s）来作最小分段，若有 200 个节拍的乐曲，则必须使用一个含 200 个或以上存储单元的 ROM 来作乐谱存储器。

按照以上的讨论，如果有一个存储了 200 单元乐谱码的 ROM（即模块 MUSIC_DATA），则必须有一个计数器为此 ROM 产生地址信号。这个计数器必须是 200 进制的（如果需要连续反复演奏），且此计数器的时钟必须与图 12-17 中 MUSIC_DATA 的地址锁存时钟 inclock 的输入频率一致，这里可以是 4Hz。这个计数器就是模块 CNTSTEP。图 12-17 中的 CNTSTEP 是一个 138 进制的计数器，这说明，放置于 MUSIC_DATA 中的这首曲的音符数是 138 个，换言之，演奏完这首曲，要 138 个节拍，总时间＝0.25s×138＝34.5s。

模块 CNTSTEP 的参数设置界面如图 12-18 所示，其中选择模为 138 的加法计数方式，即选择“Modulus，with a count modulus of 138”。

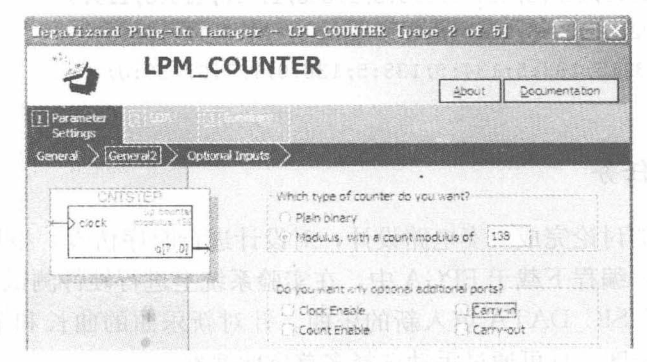


图 12-18 模块 CNTSTEP 的参数设置界面

2. 电路设计与数据文件生成

根据图 12-17 所示的乐曲自动演奏电路，设演奏乐曲是“梁祝”，故其乐谱码 mif 文件如例 12-1 所示，共 138 个节拍（乐谱码）。所以需要有一个大于此存储量的 ROM。此 ROM 即 MUSIC_DATA。它的输出数据位宽是 4，地址线宽是 8，所以有 256 个存储单元。用 Quartus Ⅱ编译时，将自动把文件 data1. mif 配置给 LPM_ROM：MUSIC_DATA。

由 MUSIC_DATA 输出的乐谱码与 INX2CODE 输入值的关系是，低 8 度简谱 0、1、2、3、4、5、6、7（0 是休止符）对应为 0、1、2、3、4、5、6、7；中 8 度简谱 1、2、3、4、5、6、7 对应为 8、9、10、11、12、13、14；高 8 度简谱 1 对应为 15 等。

电路中的 CLK4HZ 输入 4Hz 时钟频率。

如果要演奏其他乐曲，必须首先给出例 12-1 的乐曲乐谱码 mif 文件。再根据此文件的最大乐谱码数，设计一个对应进制的计数器作为乐谱码存储 ROM 的地址发生器。

【例 12-1】

```
WIDTH =4;--“梁祝”乐曲乐谱码 mif 文件。设文件名:data1.mif
DEPTH =256;
ADDRESS_RADIX =DEC;
DATA_RADIX =DEC;
CONTENT  BEGIN--注意实用文件中要展开以下数据,每一组占一行
00:3;01:3;02:3;03:3;04:5;05:5;06:5;07:6;08:8;09:8;
10:8;11:9;12:6;13:8;14:5;15:5;16:12;17:12;18:12;19:15;
20:13;21:12;22:10;23:12;24:9;25:9;26:9;27:9;28:9;29:9;
30:9;31:0;32:9;33:9;34:9;35:10;36:7;37:7;38:6;39:6;
40:5;41:5;42:5;43:6;44:8;45:8;46:9;47:9;48:3;49:3;
50:8;51:8;52:6;53:5;54:6;55:8;56:5;57:5;58:5;59:5;
60:5;61:5;62:5;63:5;64:10;65:10;66:10;67:12;68:7;69:7;
70:9;71:9;72:6;73:8;74:5;75:5;76:5;77:5;78:5;79:5;
80:3;81:5;82:3;83:3;84:5;85:6;86:7;87:9;88:6;89:6;
90:6;91:6;92:6;93:6;94:5;95:6;96:8;97:8;98:8;99:9;
100:12;101:12;102:12;103:10;104:9;105:9;106:10;107:9;108:8;109:8;
110:6;111:5;112:3;113:3;114:3;115:3;116:8;117:8;118:8;119:8;
120:6;121:8;122:6;123:5;124:3;125:5;126:6;127:8;128:5;129:5;
130:5;131:5;132:5;133:5;134:5;135:5;136:0;137:0;138:0;
END;
```

3. 设计实践任务

（1）根据以上的讨论完成演奏电路设计，对设计进行时序仿真，根据仿真波形分析说明各模块电路特性，编程下载于 FPGA 中，在实验系统上进行硬件测试，完成实验报告。

（2）在模块 MUSIC_DATA 填入新的乐曲。针对新乐曲的曲长和节拍情况改变模块 CNTSTEP 的计数长度，且可通过手动选择多首歌曲演奏。

12.4 直流电机测控电路设计

通常用PWM（Pulse Width Modulation，脉宽调制）信号控制直流电机，利用其不同的占空比来改变电机的转速。然而，当直流电机的负载发生变化时，相同占空比的PWM并不能保证电机继续保持原有的转速。即当负载增加后，必须提高占空比，才能维持原有的转速。因此，为了稳定电机的工作状态，必须有一个闭环的控制系统，时刻检测其转速，及时调整PWM的占空比，使电机始终保持预定的转速。

1. 电路结构与原理

图12-19所示的是一个直流电机闭环控制系统模块图。直流电机的转速由一对红外发光管和接收管担任，红外光接收管只有当转盘上的小孔旋转到顶部位置时，才能接收到发光管发出的红外光线，从而在电路上输出一个脉冲。

由于红外传感器接收到的信号里含有大量随机毛刺信号，其输出信号必须通过消抖动模块进行滤波。消抖动后的信号包含了电机的转速信息，此信号将通过频率计的测试，获得电机的转速（s^{-1}）。实测的转速值，一方面输出后在数码管上显示出来，另一方面进入一个比较器，即如图12-19所示的“实测转速与预置转速比较器”。

此比较器将实测的转速与键盘输入的预设的转速进行比较，获得差值，用于控制PWM发生模块输出信号的占空比，从而使电机的转速始终跟踪上预设的转速。

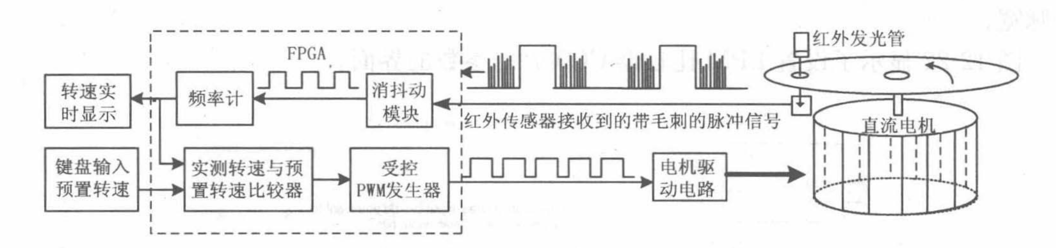


图12-19 直流电机测控电路模块图

2. PWM信号发生器的原理与设计

PWM信号发生器演示性电路如图12-20所示，其中的模块CNT8是一个8位加法计数器，模块COMP则是一个8位比较器。当此比较器的dataa端输入值大于等于datab端输入值时，输出端口ageb输出高电平，否则输出低电平。

图12-20显示，比较器的datab端接收来自计数器CNT8的输出信号，而dataa端输入一个8位常数。图12-21所示的波形演示了控制输出的方波占空比的原理。图上方的锯齿波是一个8位计数器的输出波形，最大值是255。

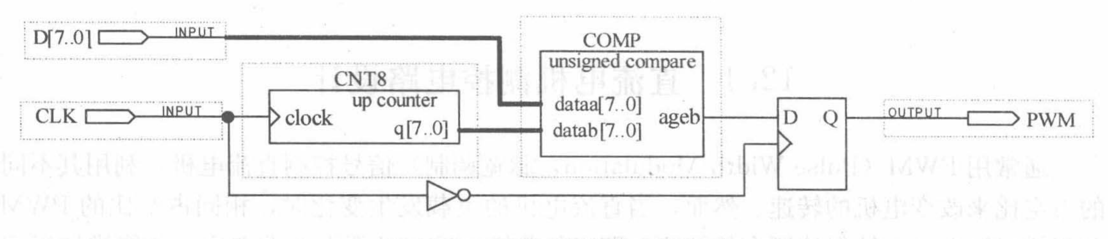


图 12-20 PWM 信号发生器电路

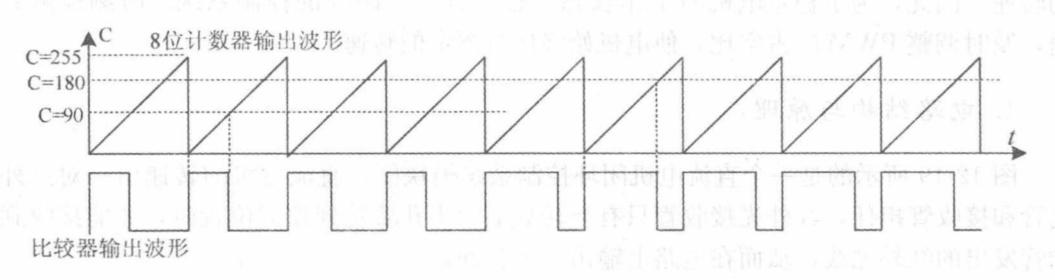


图 12-21 PWM 信号生成原理图

图 12-21 中显示，当比较器的 datab 端输入的常数 C＝90 时，当计数器输出进入比较器的值小于 90 时，比较器输出较窄脉宽的方波信号，而当进入比较器的预置常数 C＝180 时，比较器输出了更大占空比的信号，其脉宽是 C＝90 时的一倍。显然，比较器输出方波的脉宽与 dataa 端口所置的常数大小成正比，占空比越大，PWM 传输给电机的平均功率也就越大。而通过改变比较器 dataa 端的来自外部的输入数据，就能容易地改变输出的 PWM 的脉宽。

图 12-22 显示了设置 LPM 比较器 COMP 的参数的界面。

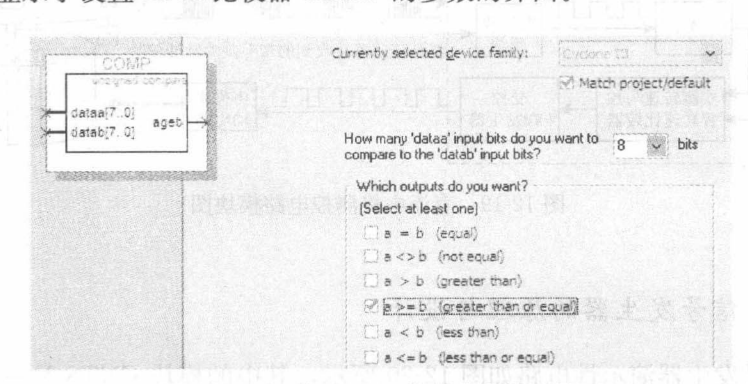


图 12-22 8 位 LPM 比较器参数设置界面

3. 消抖动模块设计

在实用数字电路设计中，这无疑是一种十分重要的功能模块。除了在之前的章节中提到过的两种消抖动电路外，还有许多其他类型的电路可用于消除信号的毛刺，比如通过对脉冲前后沿的毛刺进行计数比较的方法。但有一个共同的问题，就是在实用中需要注意消抖动电路工作时钟频率的选择。这要对信号毛刺的脉宽和频率有一个大概的估计，并由此

确定工作时钟频率的确切范围。

4. 其他模块的设计

作为一般的设计要求，图 12-19 中的频率计可直接使用本章 12.1 节设计项目的电路。但对于测定转速极低（每秒转速小于 1）的情况，就不能使用这种频率计了。因为此类频率计的缺点是，所测信号的频率越低，则测量精度越低，且根本无法测量小于1Hz 的信号频率。因此，这是一个值得读者探讨的方面。

5. 设计实践任务

（1）根据以上的讨论和图 12-19，设计出对直流电机的控制电路，要求电路在尽可能短的时间内跟踪上设置转速，以及当电机负载变化时能较好地稳定转速。

图 12-19 中的转速比较器需要读者进行自主设计。此外，键盘输入的预设转速也应该在外部的数码显示器上显示出来。

对于一般的实验用简单电机，其驱动电路比较简单，无需增加特别的电路。

（2）对于含有 A/D 或光栅等类型的转速传感器的专业直流电机，根据 12-19 所示的模型，可以设计出转速小于 1 转每秒，且含较大功率转矩的电路。这种系统适合于作精密加工，读者不妨一试。

12.5 DDS 信号发生器设计

DDS（Direct Digital Synthesizer，直接数字合成器）是一种新型的频率合成技术，具有较高的频率分辨率，可以实现快速的频率切换，并且在改变时能够保持相位的连续，很容易实现频率、相位和幅度的数控调制。因此，在现代电子系统及设备的频率源设计中，尤其在通信领域，直接数字频率合成器的应用尤为广泛。

1. DDS 原理

传统的生成正弦波的数字方法如图 10-23 所示，即利用 ROM 和 DAC，再加上地址发生计数器和寄存器即可。在 ROM 中，每个地址对应的单元中的内容（数据）都相应于正弦波的离散采样值，ROM 中必须包含完整的正弦波采样值，而且还要注意避免在按地址读取 ROM 内容时可能引起的不连续点，避免量化噪音集中于基频的谐波上。

当时钟频率 f_{clk} 输入地址发生计数器时，地址计数器所选中的 ROM 地址的内容被锁入寄存器，寄存器的输出经 DAC 恢复成连续信号，即由各个台阶重构的正弦波；若相位精度 n 比较大，则重构的正弦波经适当平滑后（通过滤波）失真比较小。当 f_{clk} 发生改变后，则 DAC 输出的正弦波频率就随之改变，但输出频率的改变仅决定于 f_{clk} 的改变。

为了控制输出频率更方便，可以采用相位累加器，使输出频率正比于时钟频率和相位增量之积。图 12-23 采用了相位累加方法的直接数字合成系统，把正弦波在相位上的精度定为 n 位，于是分辨率相当于 $1/2^n$。用时钟频率 f_{clk} 依次读取数字相位圆周上各点，以这里的数字值作为地址，读出 ROM 中相应的值（正弦波的幅度），然后经 DAC 重构正弦

波。这里比图 10-23 的简单系统多了一个相位累加器，它的作用是在读取数字相位圆周上各点时可以每隔 FWD 个点读一个数值，FWD 即为图 12-23 中的频率字，这样，DAC 输出的正弦波频率 f_{sin} 就等于基频 $f_{clk}/2^n$ 的 FWD 倍，即 DAC 输出的正弦波的频率满足：

$$f_{sin} = \mathrm{FWD}(f_{clk}/2^n) \tag{12-1}$$

式中的 f_{clk} 是 DDS 系统的工作时钟，即图 12-23 中的寄存器系统时钟。通常，式（12-1）的 n 可取值在 24～32 之间。由图 12-24 可知，其相位分辨率至少是 1/16777216，相当于 2.146×10^{-5} 度。相位增量值可预置。通过相位累加器，选取 ROM 的地址时，可以间隔选通。

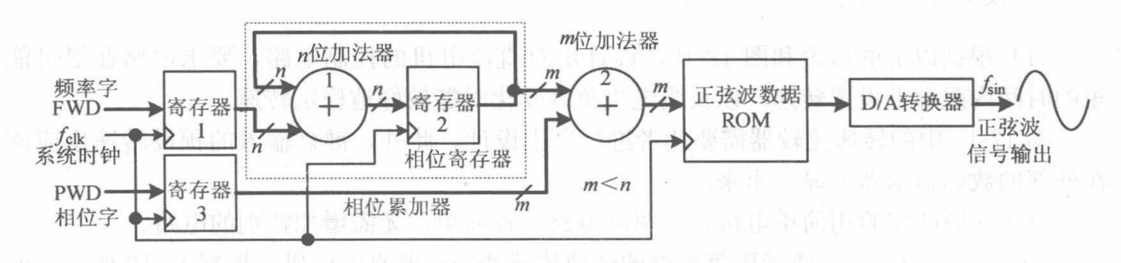
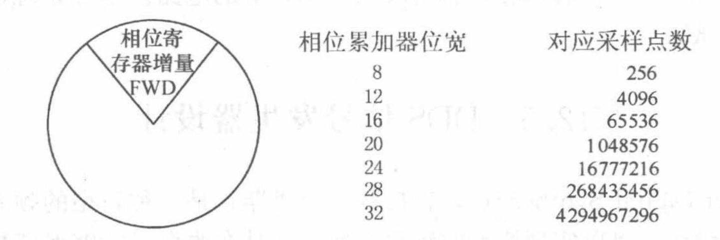


图 12-23　DDS 基本原理组成框图



相位累加器位宽	对应采样点数
8	256
12	4096
16	65536
20	1048576
24	16777216
28	268435456
32	4294967296

图 12-24　相位累加器位宽和采样点关系

图 12-23 中的 m 通常是 10～16 位，是为了减少 ROM 的容量；这里，若 DAC 的位数为 m 位，则所用 ROM 的字长也为 m。m 是对 n 的截断获得的，是取 n 位的最高 m 位。

如图 12-23 所示的 DDS 基本原理组成框图结构中，f_{clk} 来自高稳定性晶振，或由锁相环提供，用于提供 DDS 中各种部件的同步工作。DDS 核心的相位累加器由一个 n 位字长的二进制加法器和一个有时钟 f_{clk} 取样的 n 位寄存器（寄存器 2）组成，作用是对频率控制字 FWD 进行线性累加；正弦波型数据存储器中所对应的是一张函数波形查询表，对应不同的相位码址输出不同的幅度编码。

对于图 12-23 所示的 DDS，当相位控制字 PWD 为 0 时，相位累加输出的序列对波形存储器寻址，得到一系列离散的幅度编码。该幅度编码经 D/A 转换后得到对应的阶梯波，最后经低通滤波器（高频情况下无需专门的滤波电路，可通过电路本身的分布阻容进行滤波）平滑后可得到所需的模拟波形。相位累加器在基准时钟的作用下，进行线性相位累加，当相位累加器加满量时就会产生一次溢出，这样就完成了一个周期，这个周期也就是 DDS 信号的一个频率周期。

DDS 的特点可归纳如下：

(1) DDS 的频率分辨率在相位累加器的位数 n 足够大时，理论上可以获得相应的分辨

精度，这是传统方法难以实现的。

(2) DDS是一个全数字结构的开环系统，无反馈环节，因此其速度极快，一般在纳秒量级（主要取决于FPGA的速度）。

(3) DDS的相位误差主要依赖于时钟的相位特性，相位误差小。此外DDS的相位是连续变化的，形成的信号具有良好的频谱，传统的直接频率合成方法无法实现。

2. DDS信号发生器设计

图12-25是根据图12-23的基本DDS原理框图作出的具体电路原理图的顶层设计，其中相位累加器的位宽取$n=32$。图中共有3个元件模块，说明如下：

(1) 32位加法器ADDER32B。由LPM的加/减算术模块LPM_ADD_SUB构成，设置了流水线结构，使其在时钟控制下有更高的运算速度和输入数据稳定性。加法器ADDER32B的参数设置界面如图12-26所示。

(2) 32位寄存器DFF32。由LPM_FF宏模块担任。ADDER32B与DFF32构成一个32位相位累加器，其高10位PA[31..22]作为波形数据存储器SIN_ROM的地址。DFF32的参数设置界面如图12-27所示。

(3) 正弦波形数据存储器SIN_ROM。正弦波形数据ROM模块SIN_ROM的地址线和数据线位宽都是10位。这就是说，其中的一个周期的正弦波数据有1024个，每个数据有10位。其输出可以接一个10位的高速DAC；若是8位的DAC1832，可截去低2位输出。

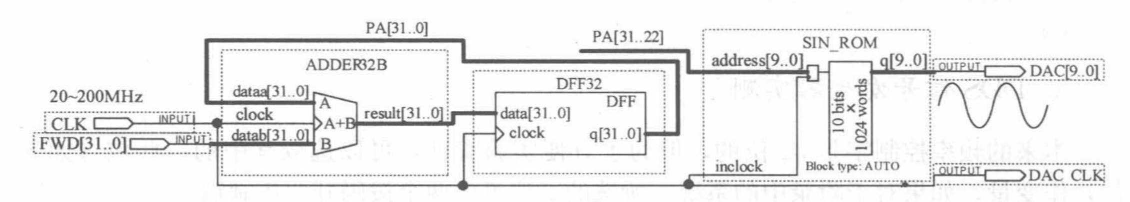


图12-25 DDS信号发生器电路顶层原理图

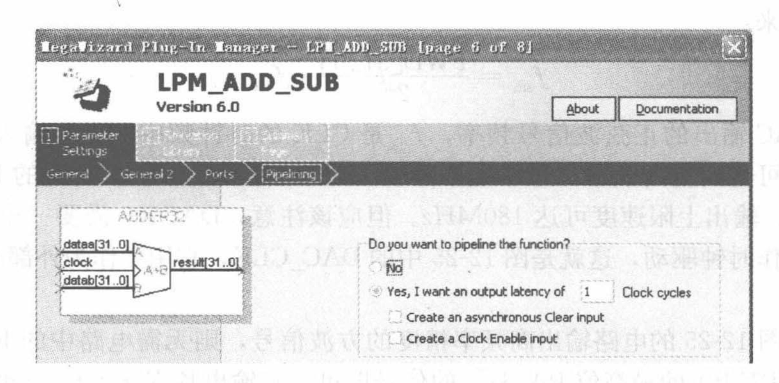


图12-26 设置LPM加法器参数界面（设置流水线结构）

作为此ROM的初始化配置文件如例12-2所示。完整的数据获得方式很多，如用C程序生成，或用MATLAB生成，也可以参考附录使用mif文件生成器生成。

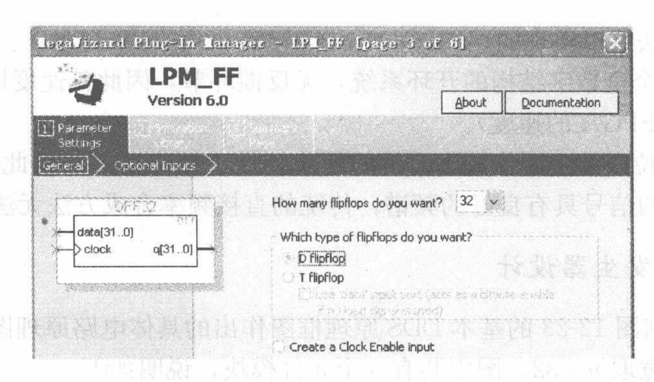


图 12-27　LPM_FF 寄存器设置界面

【例 12-2】 10 位正弦波数据文件 rom_data.mif。

```
WIDTH =10;
DEPTH =1024;
ADDRESS_RADIX =DEC;
DATA_RADIX =DEC;
CONTENT BEGIN
  0:513;1:515;2:518;3:521;4:524;5:527; 6:530;7:533;
  8:537;9:540;10:543;11:546;13:549;13:552;14:555;
  ......(略去部分数据)
  1018:493;1019:496;1020:499;1021:502;1022:505;1023:508;
END;
```

3. DDS 信号发生器实测

本来的频率控制字是 32 位的，但为了方便实验验证，可以选取其中的一些位（如 8 位）作变量。如果对于附录中的系统，所选的 8 位可由两个拨码开关控制输入。

频率控制字 FWD[31..0] 与由 DAC[9..0] 驱动的 DAC 的正弦信号的频率的关系，可以由下式算出来：

$$f_{\sin}=\frac{\text{FWD[31..0]}}{2^{32}}\cdot f_{\text{clk}} \tag{12-2}$$

其中 $f_{\sin}$ 为 DAC 输出的正弦波信号频率，f_{clk} 是 CLK 的时钟频率，直接输入是 20MHz，接入锁相环后可达到更高频率。频率上限要看 DAC 的速度。如果接高速的 DAC，如 10 位的 DAC900，输出上限速度可达 180MHz。但应该注意，DAC900 需要一个与数据输入频率相同的工作时钟驱动，这就是图 12-25 中的 DAC_CLK，它用于作为外部 DAC 的工作时钟。

如果希望图 12-25 的电路输出高频率精度的方波信号，则无需电路中的 ROM 和外部的 DAC 了，只需 PA 的最高位 PA[31] 的信号即可，其输出频率 $f=f_{\sin}$。此外，由于不受 DAC 速度的限制，输出的方波频率 f 可以远高于 $f_{\sin}$。

图 12-28 是图 12-25 电路的仿真波形。尽管这个波形只是局部的，但也能看出 DDS 的部分性能。即随着频率字 FWD 的减小，电路中 ROM 的数据输出的速度也随之减低。如

当 FWD=0000FB00H、…、00006A 时，DAC 输出数据的速度有很大不同。

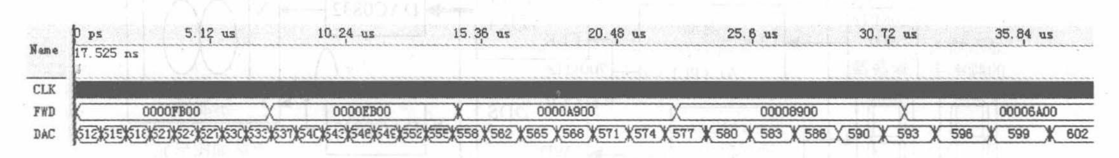


图 12-28 图 12-25 所示电路的仿真波形

如果外部 DAC 是 DAC0832，只需将 DAC[9..2] 输出给 0832 即可，信号频率算法不变。但要注意 0832 的速度只有 1MHz。

4. 设计实践任务

(1) 利用附录 mif 生成软件生成 10 位二进制数的正弦信号波形数据，有 1024 个点，按照以上的讨论，设计 DDS 信号发生器，给出时序分析结果，利用实验系统上的 DAC 进行硬件验证，同时验证式 (12-2)。

(2) 不用波形数据 ROM 模块和 DAC，直接将相位累加器的最高位 PA[31] 输出，验证式 (12-2)。

(3) 设计一个方波信号发生器。要求输出频率范围 50Hz～400kHz，且在这个区域内，输出信号频率的最大步进小于 3Hz；要求占控比可数控，精度为 1%，首先计算出此信号源的 DDS 的相位精度 n，并在此基础上设计信号发生器。

提示：可数控占空比功能设计可参考 12.4 节的 PWM 设计原理。

(4) 设计一个锯齿波和三角波信号发生器。要求输出频率范围 50Hz～300kHz，且在这个区域内，输出信号频率精度是 4Hz。计算出此信号源的 DDS 的相位精度 n，并在此基础上设计信号发生器。

提示：对于锯齿波可以直接截取图 12-24 中的 PA 的最高几位，如 PA[31..22]，输出给 DAC，不再需要数据 ROM。对应三角波，则需要单独设计一个模块，输入信号是被截取的 PA 的高几位。三角波发生模块的功能是这样设计的，首先使此模块的输出信号在开始时，是按照 PA 原有的变化进行递增，在到达其最大值的 1/2 时再进行递减，直至为 0。如此往复而输出的三角波形的频率计算公式仍是式 (12-2)。

(5) 使用其他形式的 DAC 取代 DAC0832：

① 选择 10 位并行数据输入高速 DAC，如 DAC900 或 5651，使输出的波形有更高的频率，更具有实用性。

注意： 此类 DAC 需要有一个同步的时钟信号，即图 12-25 的 DAC_CLK。

② 选择串行 DAC (如 TLC5615、TLV5637 等)。对于此类 DAC，必须为其设计一个用于控制的状态机。

(6) 根据图 12-29 的设计方案，设计一个李萨如图信号发生器。要求键盘能为 FPGA 中的两个独立的 DDS 模块分别输入频率字，并显示出来。两路 DAC 可以使用 DAC0832。

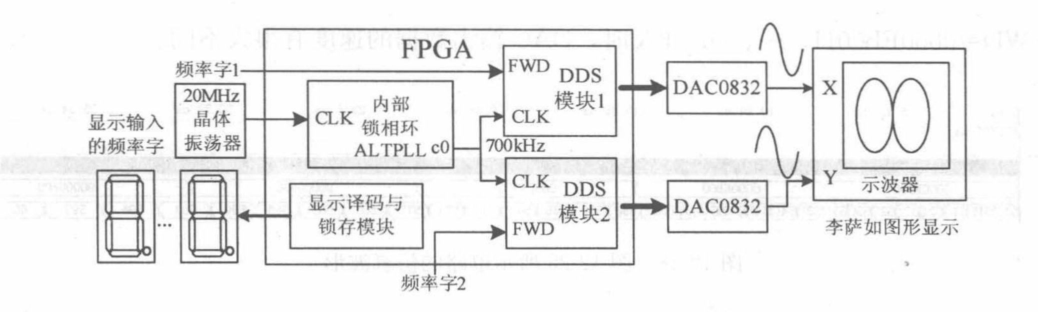


图 12-29　李萨如图信号发生器结构框图

12.6　数字移相信号发生器设计

数字移相信号发生器曾经是全国大学生电子设计竞赛赛题中的一个部分，其核心结构就是 DDS 信号发生器。以下介绍其设计原理与实现方法。

1. 电路结构与原理

将图 12-25 的电路稍加改变就能构成一个数字移相信号发生器。电路结构如图 12-30 所示，仍然是 DDS 的基本工作原理。电路的上半部结构与图 12-25 相同，是一个基本 DDS 信号发生器，而电路的下半部通过一个 10 位加法器 ADDER10、一个 10 位寄存器 DFF10 和一个相同的正弦波形数据存储器 SIN_ROM，输出另一路正弦信号数据。10 位加法器的一端接与上方的 ROM 相同的地址信号值 A[31..22]；另一端接一个输入常数 PWD[9..0]，即相位控制字，它的大小与输出的 DACF[9..0] 和 DACP[9..0] 构成的正弦波间的相位差成正比。因此控制 PWD[9..0] 的大小，就能线性地控制两个输出的正弦波的相位差。其原理可参考图 12-23 及其说明。

显然，图 12-30 的电路用了一个相位累加器，输出的这两路信号频率相同，频率步进同步，但这两路信号间的相位差可通过相位字 PWD 进行数控。相位控制的精度与 PWD 的位宽相关。

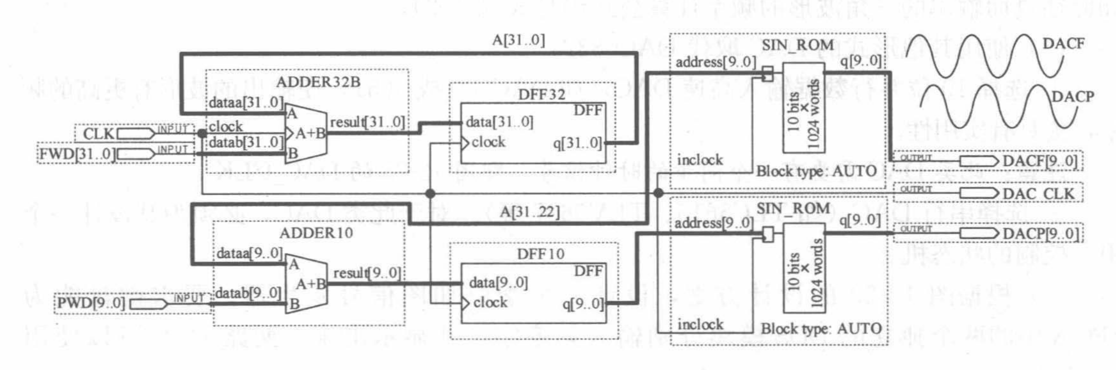


图 12-30　数字移相信号发生器电路模型图

2. 设计实践任务

根据以上的结构和原理描述，设计数字移相信号发生器。注意实验验证中要用到双DAC，并且要安排两组控制键，一组向FPGA中的DDS模块输入频率控制字，进行频率数控；另一组进行相位数控。根据相位精度n计算频率数控，并计算相位数控的精度。利用实验系统上的双DAC进行硬件验证，在双踪示波器上显示波形。

详细说明设计原理、电路功能及时序特点，并完成实验报告。

12.7 简易数字电压表设计

本项设计融合了数字电路技术中的多个方面的内容，有一定的综合性，包括A/D的使用、状态机的设计、存储器的使用、显示器的控制等。同时也为下一个设计项目做准备。

1. 电路结构与原理

电路如图12-31所示。ADC使用0809，被测电压通过一个电位器进入0809的IN0端。0809对此电位器输出的电压值进行采样受控于FPGA中的状态机。0809采样变换后的8位数据进入FPGA中，但考虑到0809的工作电压是+5V，而FPGA的I/O端口电压是3.3V，所以，0809输入FPGA的8位数据线都必须分别串接300Ω左右的限流电阻。

图12-31中的状态机的工作时钟可来自内部锁相环，频率为3MHz左右。状态机的输出信号LOCK控制一个8位寄存器，锁存来自0809变换好的数据。这个数据通过一个计算查表LPM_ROM，将寄存器中的数据折算为电压值。这个电压值再通过显示仪译码器在两位数码上显示出来，精度约0.1V。

内部锁相环还为0809提供一个500kHz的工作时钟。

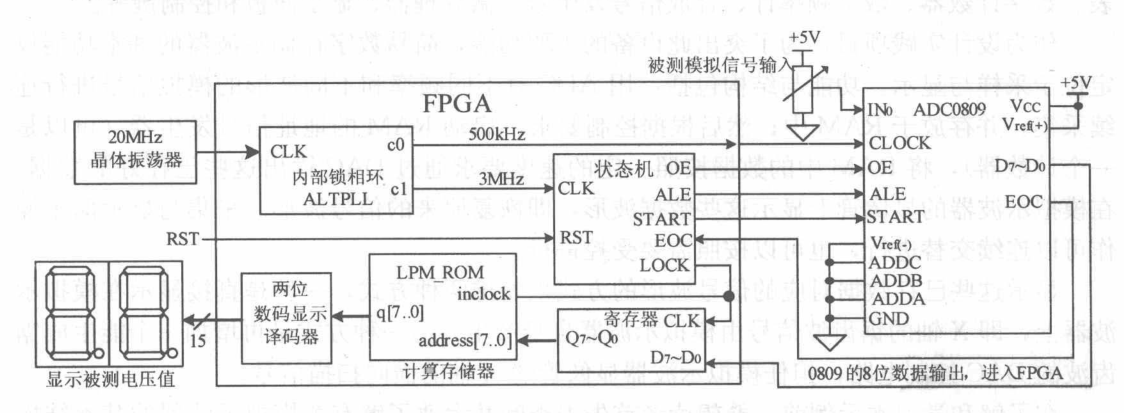


图12-31 简易数字电压表电路模型图

2. 设计实践任务

(1) 根据图12-31的电路结构，设计一个数字电压表，精度是0.1V。

(2) 根据图12-31的电路结构，设计一个数字电压表，但作一点改进，即将图中的模块LPM_ROM和两位数码显示译码器及其它们的功能合并起来，用一个新的LPM_ROM替代，输入这个ROM的地址线仍是8位，但输出的数据宽度是15位。其中的两个7位分别接外部的7段显示数码管，而最高位接小数点。即此ROM的任务包含了数据转换和两位7段显示译码。对此设计，需要编辑一个C程序，为ROM中的查表数据计算和表格建立，即对于可能输入的256个数值中的每一个值计算出对应的电压值，再转换成对应的两个7段显示码（有14或15位）。然后将所有的数据按照mif文件格式编辑生成，并输出一个标准文件，供系统综合时配置于LPM_ROM。

(3) 将图12-31中的ADC换成10位或16位精度更高的ADC，但必须是串行输出型ADC（如MAXIM187/189、ADS1100、ADS7816、TLV2541、TLV1572等）。这种情况下，需要重新设计控制ADC的状态机，且显示用数码管则根据情况增加1至多个。

(4) 将图12-31中的寄存器换成LPM_RAM，用以记录一段时间内的输入电压的变化。并能通过外部控制，在数码管上显示出曾经记录的数据。

12.8 简易数字存储示波器设计

简易数字存储示波器也曾经是全国大学生电子设计竞赛赛题。其基本功能就是利用ADC将模拟信号采集进来暂存于RAM中，然后依照一定的要求，通过DAC在显示器上重现这些波形，以备分析研究。当然也可以不用DAC，而直接在计算机上或其他显示设备上显示，如单片机加液晶显示器等。

模拟示波器只能在当时显示被采样的信号波形，当这些信号消失后，就无法再显示和分析这些信号了，对此，数字存储示波器的优势就十分明显。功能完整的数字存储示波器的组成比较复杂，它包括性能良好的ADC、DAC及其控制系统、数字电位器、数字电压表、数字计数器、数字频率计、合成信号发生器、微处理器、显示面板和控制键等。

作为设计实践项目，为了突出此设备的主要功能，简易数字存储示波器的基本功能仅定位于采样与显示。功能与结构包括：用ADC对不同频率和不同波形的模拟信号进行连续采集，并存放于RAM中；然后根据控制要求，启动RAM的地址信号发生器（可以是一个计数器），将RAM中的数据按照一定的速度要求通过DAC输出这些已存好的数据，在模拟示波器的显像管上显示这些数据波形，即恢复原来的信号波形。采集与显示两个操作可以连续交替进行，也可以按照需要受控进行。

显示这些已存数据对应的信号波形的方式也安排两种方式，一种是直接显示在模拟示波器上，即X轴的锯齿波信号由模拟示波器自身产生；另一种方式是再增加一个能生成锯齿波的DAC控制电路，担任模拟示波器显像管的X轴的横向扫描信号。

在了解和学习本示例前，希望读者首先去查阅并大致了解有关模拟示波器的基本结构和工作原理，以及了解一些基本的使用方法。

1. 电路结构与原理

图12-32是此简易数字存储示波器的结构框图。其中的状态机担任对ADC0809采样

控制，LPM_RAM用于连续存储0809转换好的数据，其数据宽度是8，地址位宽为10。为了有效地控制此RAM，使之成为既能存储采样数据，又能成为通过DAC恢复所存数据的信号波形存储器，就必须为之增加两个控制元件，其中之一是此RAM的地址信号发生器，即一个10位二进制计数器，数据宽度恰好与RAM的地址线宽度相等，计数输出直接与RAM的地址端口相接。计数器采用LPM_COUNTER模块。而另一个是多路选择器。

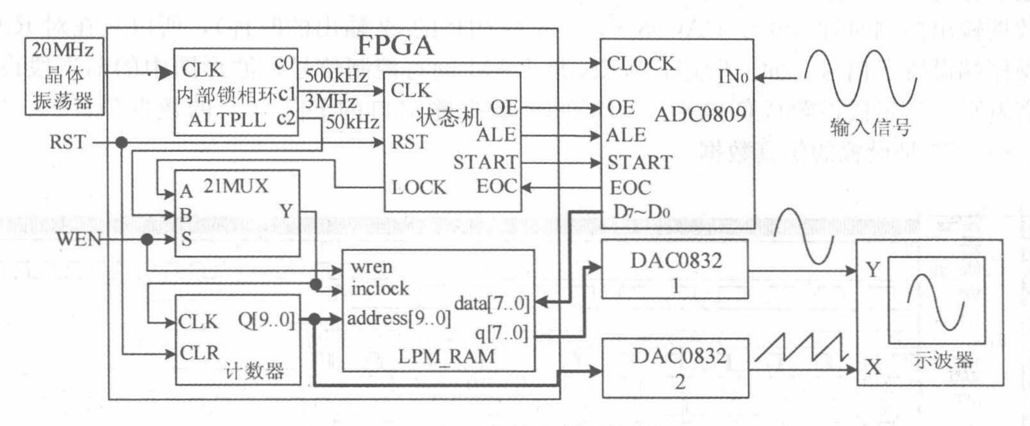


图12-32 简易存储示波器结构框图

为了控制进入计数器不同频率的时钟，以便使RAM适应两个不同的角色，即数据存储RAM和输出波形数据ROM，必须增加一个2选1多路选择器21MUX。其输入口A端接状态机输出的LOCK；B端接来自锁相环的频率输出，或接DAC的工作时钟DAC_CLK；选择端S接WEN，这是一个RAM写允许控制信号，高电平有效。

WEN=1是采样存储周期，这时电路控制ADC0809对外部信号波形进行采样。这时一方面允许RAM数据写入，即允许来自ADC0809的转换数据被存入RAM中；另一方面，使得多路选择器的S=1，即允许状态机的LOCK锁存脉冲成为RAM的地址锁存脉冲。即每来一个LOCK脉冲，将使计数器CNT10B递增1，使RAM的地址增1，同时将采入的数据D[7..0]被LOCK锁入当前的地址。即此时存一个数据进入RAM8对应的地址单元。

由此可知，这时被存入RAM中的数据一定是刚采样转换好的数据。在采样周期，即WEN为1的时间长度应该与ADC0809的采样速度及RAM的容量有关。

WEN=0是波形显示周期，即将刚才存入RAM的波形数据通过DAC显示出来的周期。此时，一方面RAM的数据写入被禁止，只允许数据读出，类似一个ROM；另一方面DAC工作时钟控制（如果是DAC0832，可以不用此时钟）驱动计数器，产生RAM地址，此时RAM的数据随着此时钟驱动，将存入的数据通过数据输出口Q[7..0]进入DAC0832的数据通道，从而能在示波器上显示并恢复出刚才被ADC采样信号的波形。

注意：DAC_CLK的频率与DAC的工作频率相关。对于DAC0832，此频率不高于1MHz。而状态机工作时钟FM_CLK的频率可以比较高（1～150MHz），这里取3MHz。

2. 时序分析

当根据结构图 12-32 设计编辑好对应的电路后，首先必须进行仿真测试，图 12-33 就是此对应的时序仿真波形图。设 FM_CLK 是状态机工作时钟，从图中可以看出，设定的频率比较高。RST 是状态机的寄存器以及作为地址发生器的计数器的清 0 控制信号，都是低电平有效。图中有两次清 0 控制，一次是采样前，另一次是波形数据输出前。DAC_CLK 是数据输出控制时钟（对于 DAC0809，则是锁相环的 c2 输出的时钟），所以只在对 RAM 读操作端设置了信号。如上所述，WEN 是 RAM 读写控制信号，波形图中的前半段设置成高电平；后半段设置成低电平。ADC0809 数据输出口的 D[7..0] 的数据值：24、28、30、…、50 是设置的仿真数据。

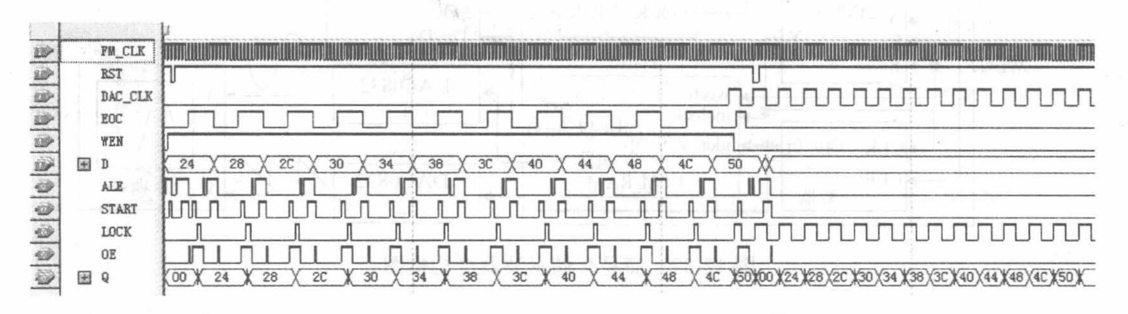


图 12-33 简易存储示波器仿真波形

其他的信号，包括 EOC、ALE、START、OE、LOCK 等的波形特点和功能，与 ADC0809 采样控制波形图的情况完全一样。

注意 ADC 输出数据 D[7..0] 和 RAM 输出数据 Q[7..0] 的数据变化情况和时序关系。刚才提到，D 的一系列数据是人为设置的 ADC 转换好的数据，而在 WEN=1 阶段，Q 输出的数据与 D 的数据相对应，但在时间上都是在 LOCK 的上升沿后出现。说明此时的 Q 的数据来自 ADC 转换好的数据，且已被储存进 LPM_RAM 中了。

而在 WEN=0 阶段，随着 DAC_CLK 的节奏，RAM 通过 Q 将存储的数据一一输出。这时由于地址计数器已被清 0，所以输出的数据顺序与写入的数据顺序相同，也是 24、28、30、…、50。

3. 硬件测试

硬件测试将最终证明电路的可行性。实验验证中必须注意：考虑到 ADC0809 的输入电平情况（0～5V）和速度特点，需控制输入信号的频率和幅度。注意选择信号输出不要大于 5V；作为 0809 采样信号，可来自函数信号发生器的输出信号，其波形类型可以作一些选择，如正弦波、锯齿波、三角波、方波、梯形波等，但频率必须控制在 200～300Hz 间。

实测时，可以利用双踪示波器的特点，将被采样信号的波形与 DAC 输出的波形同时显示出来比较。同时利用 Quartus Ⅱ 的在系统存储器内容编辑器 In-System Memory Content Editor 实时观察对波形数据的采样情况。

4. 设计实践任务

(1) 根据图12-32，以及对其的讨论和分析，设计一个简易数字存储示波器。在实验系统上进行硬件验证，对实验系统上配置的DDS函数信号发生器产生的不同波形和不同频率的信号进行采样（幅度调谐于4～5V间），并用双踪示波器观察存储的波形。

(2) 电路能产生锯齿波信号，用以控制示波器的X信号端，以便通过利用X-Y的选择，直接控制示波器显像管来显示输出的波形信号。

(3) 用高速高精度ADC取代ADC0809，进一步完善设计。

详细说明设计原理、电路功能及时序特点，并完成实验报告。

12.9 移位相加型8位硬件乘法器设计

硬件乘法器的优势就是运算速度高。高速硬件乘法器在现代数字系统中应用广泛，如函数信号发生器、数字通信系统、数字信号处理、图像识别、工业实时控制系统中都会用到。本项目是要设计一个8位乘8位的硬件乘法器。

1. 结构与原理

为了节省逻辑资源，该乘法器是由8位加法器构成的以时序方式工作的8位乘法器。乘法通过逐项移位相加原理来实现。基本的乘法运算就是从被乘数的最低位开始，若为1，则乘数左移后与上一次的和相加；若为0，左移后以全0相加，直至被乘数的最高位。从逻辑电路图12-34及其乘法操作的时序图12-25（设相乘数为9FH和FDH）上可以清楚地看出此乘法器的工作原理。时序波形图12-35中，START信号的上跳沿及其高电平有如下两个功能（它的低电平则作为乘法使能信号）：

(1) 对9位寄存器REG9B和7位右移寄存器SHFT_R7清0。

(2) 将被乘数B[7..0]向右移移位寄存器REGS_R8加载。

CLK为乘法时钟信号。当被乘数被加载于8位右移寄存器REGS_R8后，随着每一时钟节拍，最低位在前，由低位至高位逐位移出。

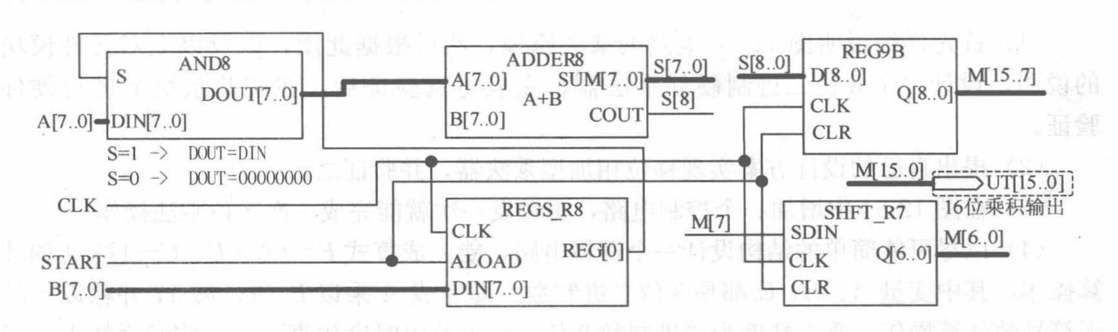


图12-34 8位乘法器逻辑原理与结构图

当右移寄存器REGS_R8中的被乘数右移出的最低位为1时，模块AND8的通道打

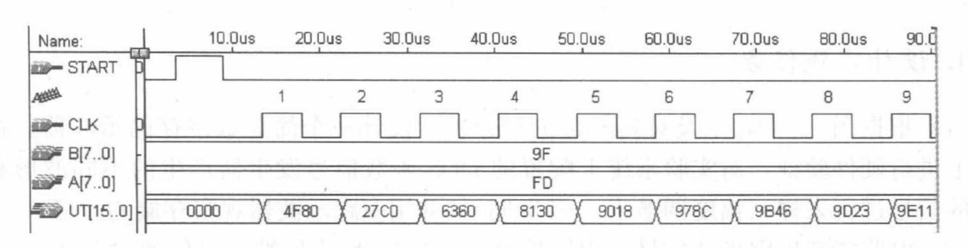


图 12-35 8 位移位相加乘法器运算逻辑波形图

开，8 位乘数 A[7..0] 在同一节拍进入 8 位加法器 ADDER8，与上一次锁存在 9 位锁存器 REG9B 中的高 8 位进行相加，其和在下一时钟节拍的上升沿被锁进此锁存器；此时，在同一时钟节拍下，REG9B 输出的最低位 M[7] 被移入 7 位右移移位寄存器 SHFT_R7 的最高位，而其他 6 位顺序向右移。

当右移寄存器 REGS_R8 中的被乘数右移出的最低位为 0 时，此 AND8 全 0 输出。

如此往复，直至 8 个时钟脉冲后，最后乘积的高 9 位出现在 REG9B 的输出端口，而低 7 位出现在 SHFT_R7 的输出端口。

从图 12-35 可见，当 9FH 和 FDH 相乘时，第 1 个时钟上升沿后，其移位相加的结果是 4F80H，第 8 个时钟上升沿后，最终相乘结果是 9D23H，即：9FH×FDH=9D23H。

此项设计的元件设计说明：

- 模块 AND8。AND8 相当于一个 1 位乘法器，其功能类似于一个特殊的与门，即当输入 S 为 1 时，DOUT 直接输出 DIN；而当 S 为 0 时，DOUT 输出 00000000。因此，AND8 本质上就是一个广义译码器，可用简单的 case 语句描述。
- 模块 ADDER8。ADDER8 是一个 8 位无符号数加法器，COUT 是进位输出。获得 ADDER8 的最简单的方法是使用 LPM 模块。
- 模块 REG9B。这是一个普通寄存器，可用 9 个 D 触发器构建。
- 模块 REGS_R8 和 SHFT_R7。REGS_R8 和 SHFT_R7 都是右移移位寄存器，但是，前者是 8 位并进串出型，而后者是 7 位串进并出型。它们都可用 D 触发器构建。

2. 设计实践任务

(1) 首先详细说明图 12-34 电路的乘法原理，然后根据此图，以及以上对元件模块的说明，设计一个 8 位二进制硬件乘法器，完成仿真验证后，在实验系统上进行硬件验证。

(2) 提出自己的设计方案实现移位相加型乘法器，并验证之。

(3) 在图 12-34 中增加一个控制电路，每触发一次就能完成一次 8 位乘法操作。

(4) 以尽可能简单的结构设计一个逻辑电路，能完成算式 $F=(A\times B+C-17)/4$ 的计算操作，其中变量 A、B、C 都是 8 位二进制数。即完成 A 乘以 B 加 C 减 17 并除以 4 的无符号数计算操作，要求精度为二进制数 8 位。首先给出时序仿真，再在实验系统上进行硬件验证。最后给出一个尽可能高速完成此算式的逻辑电路设计方案。

提示：除以 4 的操作可以用移位的方式完成。

12.10 基于状态机的实用数字系统设计

首先设计逻辑笔。传统逻辑笔只能测试出高/低电平和脉冲信号，这远不能适应实际的需要，以下介绍使用状态机来控制逻辑笔的测试，使逻辑笔具有一定的智能化。

1. 电路结构与原理

图 12-36 所示的是智能逻辑笔结构框图，其中的测试电路模块内部的电路原理图如图 12-37（a）所示；图 12-36 左侧给出了逻辑笔测试结果的显示形式，即当数码管分别显示 H、L、Π、P、E 时，将表示所测试的结果分别是：高电平、低电平、高阻态、脉冲电平和错误电平。所谓错误电平，或称无效电平（图 2-33 中标的无效电平），即所测电平处于定义为高电平的最低值和低电平的最高值之间的电平。逻辑器件的输入输出端口，若处于这个电平，则相关的器件和电路必定存在这样或那样的问题，在此姑且称为“错误电平”。

图 12-36 中，FPGA 与逻辑笔的信号采样电路有 3 个端口相连接，即：Vo1、Vo2 和 TEST。Vo1 和 Vo2 输入进 FPGA；TEST 由 FPGA 输出。由图 12-37(a)可见，TEST 信号从 FPGA 端口输出通过两个 TTL 反向器和一个电阻后与比较器 LM393 的第 5 脚相连。这主要是考虑了 FPGA 输出的 3.3V I/O 电平不够高，驱动力不够的原因。

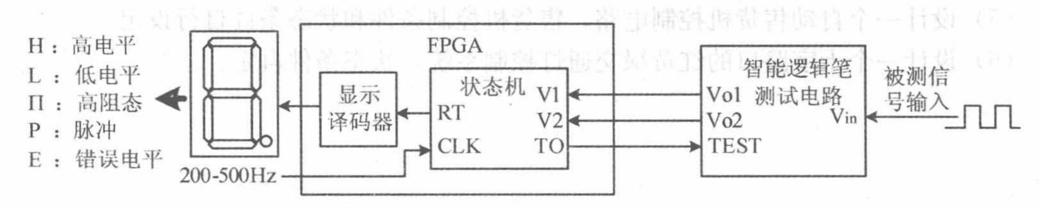


图 12-36 智能逻辑笔结构框图

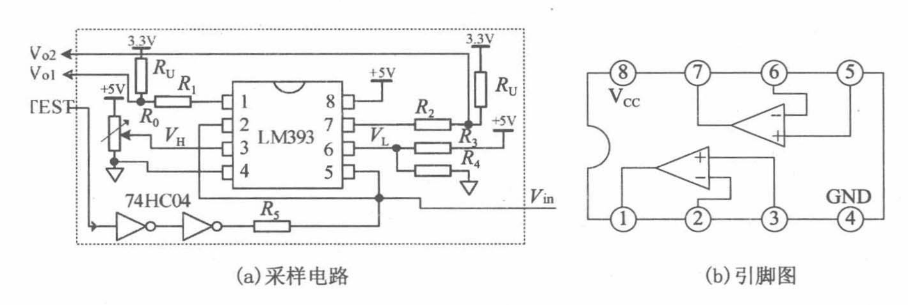


图 12-37 智能逻辑笔电平信号采样电路和 LM393 的引脚图

设计前应该首先查阅有关 LM393 的资料，它是一个双比较器。与它相接的电阻如图 12-37（a）所示，电阻 R_0、R_1、R_2、R_3、R_4、R_5 的阻值分别是：10kΩ、300Ω、300Ω、5.1kΩ、1kΩ 和 100kΩ。其中 R_0 是电位器；R_1 和 R_2 是与 FPGA 端口连接的限流电阻；R_3 和 R_4 构成了分压电阻，为 LM393 的第 6 脚提供一个参考电平 V_L，约 1V，作为被测信号标准低电平的最高值，而第 3 脚通过一个电位器获得参考电压 V_H（调谐电位器，使 V_H =3V 左右），作为被测信号标准高电平的最低值。R_5 是为测高阻态预备的，阻值高达 100kΩ。

此外，接输出电平 Vo1 和 Vo2 处的两个电阻 R_U 的阻值都是 10kΩ，是上拉电阻，接 3.3V 电平，因为 FPGA 的 I/O 高电平是 3.3V。

另外注意电平测试口 Vin 除与两个比较器的输入端 2 和 5 相连外，还通过一个 100kΩ 的电阻与 FPGA 的输出口（通过两个反相器）相接，这是测试高组态必须的电阻。因为可以通过 FPGA 的状态机对 TEST 分别输出 1 或 0，以及结合来自 Vo1 和 Vo2 测试到的电平组合，判断出测试端 Vin 的逻辑信号是高电平、低电平、错误电平、高阻态还是脉冲。

在状态机中可以这样来判断：当测到的电平 $V_{in}>V_H$，判为高电平；若 $V_{in}<V_L$，则判为低电平；若 $V_L<V_{in}<V_H$，则判为错误电平；再若当 TEST 输出 1 时判定为高电平，而 TEST 输出 0 时又被判定为低电平，则结果必为高阻态；又若高低电平不断变化，则判为连续脉冲（建议状态机的工作时钟频率不要太高，可以在 300Hz 左右）。

2. 设计实践任务

（1）根据图 12-36 和图 12-37 及以上的讨论，设计智能逻辑笔，首先自主设计图 12-37 所示的电路，然后设计用于控制的状态机，最后根据图 12-36 的形式显示出测试结果。

（2）讨论电路中的电阻 R_5 的作用，以及其阻值大小对测试结果的影响。

（3）设计一个洗衣机控制电路，洗衣机控制条件和状态条件自行设定。

（4）设计一个 4 层电梯控制电路，电梯控制条件和状态条件自行设定。

（5）设计一个自动售货机控制电路，售货机控制条件和状态条件自行设定。

（6）设计一个十字路口的红黄绿交通灯控制系统，状态条件自定。

附　录

数字技术实验系统及基本要求

根据本教材的内容安排、实验要求和希望达到的教学目标，选择与之配套的结构恰当的实验系统是十分必要的。为了实现前言中提到的既定教学目标，相比于传统教材，本书的总体内容和篇幅有较大的减少，然而实验与实践的内容却有了很大的变化和扩充，涉猎的广度和深度也有很大的提高，特别是与现代数字技术和创新理念有了巨大的拉近。这就使得在实验内容、实验方法、实验要求、实验目的和实验课时数等方面有了许多新的要求。

以下将给出配合本教材的实验规划、实验内容和实验系统的一般安排和基本要求，以及相关实验板和实验系统的主要构成与功能，以资读者参考。最后介绍 mif 文件生成软件的使用方法。

1.1　基本实验内容、方式和类型

配合本教材的实验内容可以分为 3 个部分。

1. 基于手工设计技术和通用逻辑器件的传统数字电路实验项目

这部分实验的内容、方法和实验目标与传统教材基本对应。对于本书而言，主要包括针对第 2 章、第 4 章、第 5 章（部分）、第 7 章（部分）和第 11 章的实验。

实验内容主要包括对基本逻辑器件电气性能的测试和认知、基本组合逻辑电路和时序逻辑电路的手工设计与分析；实验目标主要集中在对教材内容的实验验证上，且相比于传统教材的实验内容和学时数应该大幅减少，因为这些内容是过渡性的，不是学习的重点。

实验平台可以是传统的数字电路实验箱，基本实验器件是 74LS、74HC 和 4000、4500 等系列器件；实验方法和手段主要是手工设计、手工接插和手工测试，且以验证性实验和逻辑功能实现（基本不考虑技术指标）为主要实验内容。

2. 基于自动设计技术和 FPGA 硬件平台的基础实验项目

这部分实验项目主要包括针对第 6 章、第 7 章（部分）、第 8 章（部分）的实验。内容主要集中在通过数字系统自动设计软件和 FPGA 硬件实验平台，完成一些验证性的数字功能模块和小规模数字系统设计的实验项目；实验目标是初步掌握基于现代数字系统自动设计技术的系统建模方法、设计方法和测试方法，进而从工程实际的角度逐步深入了解数字电路的系统结构与性能特点、设计与测试的手段。这部分实验的学时数可以根据教学要

求适当增加。

从实验内容上看，这部分实验与传统实验有不少重叠，但实验方法、实验工具、实验平台，以及实验项目的实现和验证测试方法都完全不同，因为它们是基于自动设计技术的。

实验平台可以是基于 FPGA 的一切数字电路实验系统和数字逻辑自动化设计软件平台。但为了更好地适应本教材给出的教学内容、实验要求和部分性能要求及技术指标，同时更能适应现代数字系统设计技术的最新发展，建议 FPGA 硬件平台采用 CycloneⅢ系列或以上，软件平台采用 QuartusⅡ9.1 或以上。

3. 基于自动设计技术和FPGA硬件平台的自主性和综合性实验项目

这部分实验项目主要包括针对第 8 章（部分）、第 9 章、第 10 章和第 12 章的实验和设计项目。这部分实验需要占用较多的实验学时数。实验平台与以上基本相同。由于部分实验内容与现代数字技术与工程实际有更好的适应性，因此实验内容有更多的综合性；对于实验项目的要求，在实用性、自主性和创新性方面有了更多的挑战。

为此，实验环境和方式可分为两类：

(1) 基于实验室的传统实验类型。在实验室完成实验和设计项目是实现传统教学实践环节的基本途径，优势是可以利用实验室中完善的实验设备和测试工具，教师的临场指导，及同学间的互动交流；缺点是有限的实验空间，难以自行安排的实验时间，以及有限的实验课时数。这对于一些需要较多实验课时、较长前期实验准备和更灵活的实验时间安排的自主性、创新性实验项目的完成有一定的阻碍。

(2) 家庭实验类型。为了弥补以上实验类型的不足，以及基于数字系统自动设计技术的特殊性，不少学校为学习数字电路课程的学生人手配备一块基于 FPGA 的数字电路实验板，使得学生们能把一些自主性实验带回家（宿舍）去完成。这当然也得益于当前学生个人计算机普及率的大幅提高。

显然，与传统实验不同，基于自动设计技术的数字系统实验只涉及两个基本部分，即软件和硬件。前者是功能强大的设计软件，如 QuartusⅡ，这可以安装在任何个人计算机中，实验者可以在任何可能的时间中在自己的计算机上对实验项目进行建模、设计、实现、仿真、测试和验证；后者是高度集成的、硬件资源极为丰富的 FPGA 器件。由于以此器件为核心的实验板所占的空间很小，所以大多数数字系统实验可以在实验室以外的地方，包括学生宿舍中完成。这对于提高学习效率、实验效率、课余时间的利用率，以及自主创新设计项目的实现的完整性都有不可替代的好处。

1.2 数字电路实验板基本结构与功能

这里主要介绍针对数字电路自动设计技术的实验板的结构配置、功能和基本用法。

本书为了验证教材中出现的示例和完成设计项目，将理论学习与工程实践紧密结合，实验演示的硬件平台选择的是康芯公司的 KX-7C10E＋型实验板和 KX_DN8 系列模块化实验系统，它们所选择的 FPGA 分别对应 CycloneⅢ系列 FPGA：EP3C10E144 和

EP3C55F484。

如果读者手头已有类似实验系统，也同样能完成本书的实验和实训项目，只是要注意，如果目标芯片和封装不一样，须对示例中的FPGA的引脚锁定作更改，特别是对于现成的示例源文件（都是以Cyclone Ⅲ FPGA作目标器件的），则要作较多的改变，其中除引脚锁定外，还需改变LPM存储器、锁相环等。但须注意，不同系列FPGA中的锁相环的结构不尽相同，因此许多设置无法吻合，则只能放弃。早期的FPGA内没有锁相环，而Cyclone与Cyclone Ⅲ系列FPGA内的锁相环也有很大不同，例如前者的锁相环的信号输出端只有3个，且频率范围是10～270MHz，而后者有5个不同频率输出口，频率范围是2kHz～1300MHz，特别是内部嵌入式RAM规模有了前所未有的扩大。

因此作者推荐使用Altera公司较新的Cyclone Ⅲ FPGA作为实验目标器件，此外，如果读者实验板的FPGA是Cyclone IV系列的，则完全可以与Cyclone Ⅲ系列的器件全面兼容，因为这两个系列的FPGA对应相同封装情况下，除极个别引脚外，几乎可以相互通用！这就是说，可以将同封装的Cyclone Ⅲ FPGA当成Cyclone IV FPGA来使用，反之亦然。例如EP3C10E144和EP3C5E144可以与EP4CE10C22互相通用。这是因为它们的工艺（一种是60nm，另一种是65nm）接近，结构相同。这样一来，Cyclone IV器件的编译不一定需要Quartus Ⅱ 11.1了。此外，不仅仅是本教材所有硬件验证示例和实验都是基于此系列器件，更重要的是Cyclone Ⅲ FPGA的高性价比、先进的结构和较好的市场前景。此系列器件以其高集成度、高速、大规模内嵌RAM以及优秀的锁相环的性能，使之前的诸多系列，如ACEX、FLEX、APEX、Cyclone/Ⅱ等，都不可望其项背。

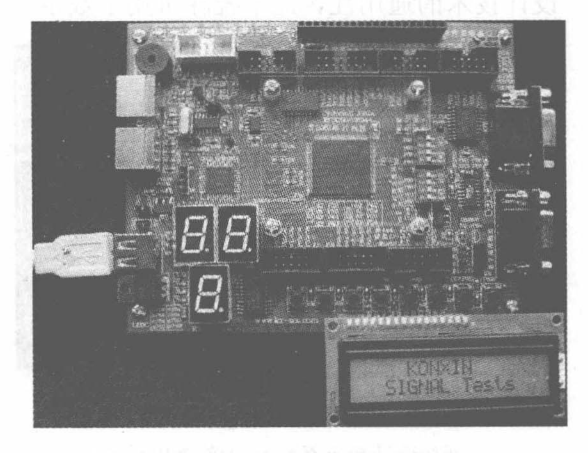
为了能更好地完成本书给出的实验设计项目，以下简要给出相关的使用说明，以备查用。

1. KX-7C10E＋系统的主要硬件配置

此系统可作为课外自主实验开发。由于KX-7C10E＋系统（图F-1）本身配置比较完整且结构紧凑，同时含许多标准接口，因此除在该系统上可完成大量实验和设计项目外，还可通过接插各类扩展模块实现更多项目的实验和创新设计。

KX-7C10E＋系统含有如下硬件配置：

- Cyclone Ⅲ型FPGA，EP3C10E144，含10 320个逻辑宏单元，2个锁相环，约90万门、43万RAM单元；FPGA配置Flash EPCS4/16（16Mb），超宽超高锁相环输出频率：1300MHz～2kHz。
- CPLD EPM3032A-44PinTQFP，1602字符液晶屏、3数码管，8发光管，1.2V、2.5V、3.3V、5V混合电压源，8键、2四位拨码开关，蜂鸣器，USB电源线，RS23通信线，4×4键盘，2个全局时钟输入口，其中一个是第2锁相环时钟口。



图F-1 KX-7C10E＋主系统板

- 标准接口系列1：VGA显示器接口，PS/2键盘接口，PS/2鼠标接口，RS232串行接口。
- 标准接口系列2：USB电源接口，JTAG编程接口，DS18B20数字温度器件接口。
- 标准接口系列3：字符型液晶接口，可接1602（2行16字符）、2004（4行20字符）、1604（4行16字符）；含中文字库64×128等液晶显示屏；点阵液晶接口，可接64×128点阵型液晶显示屏。
- 标准接口4：可接插800×480数字TFT彩色液晶屏等。
- USB-Blaster编程器与USB至RS232串行通信接口综合实验接口板。

2. KX-7C10E＋系统主要软核配置和控制模块

尽管这些IP核在本课程中尚无实验项目可以用到，但在后续课程中，如EDA技术、片上系统设计、SOPC技术、DSP、基于SOC的单片机技术、嵌入式系统等，或在一些实用创新项目实验开发中，都有可能频繁用到。

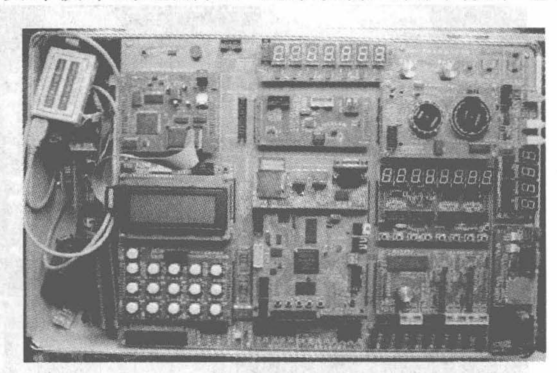
（1）8051单片机IP核：配置的51核全兼容8051CPU核，主频最高可达200MHz，是传统51单片机的十几倍，而FPGA资源仅1800个LCs，因此在10E＋系统上可实现SOC片上系统设计。

（2）8088CPU核及其他核：可运行8088CPU核构建的IBM计算机SOC片上系统，包括8088CPU核、8255I/O口扩展核、8253定时器核、8250UART串行通信核、8237DMA控制核、8259中断控制核等，以及DSP核、NCO（DDS）数控振荡器核、FIR数字滤波器、FFT等核。

（3）为了完成更多的自主设计项目，可以在KX-7C10E＋上扩展许多专用模块，如A/D模块、D/A模块、直流和步进电机模块、各类液晶模块、DDS模块、键盘鼠标模块、各类存储器模块等。

3. KX-DN8系统的主要硬件配置

功能与配置更全的KX-DN8系统（图F-2）属于实验室的配置系统，由于现代数字设计技术的通用性，此系统除可用于数字电子技术实验与设计外，也适合其他课程的实验，如EDA技术、SOC、计算机接口、CPU设计等。



图F-2 KX-DN8系列模块自由组合型创新设计数字系统综合实验开发系统

一般情况下，诸如EDA开发、计算机组成原理实验、微机原理与接口技术实验、单片机技术实验、DSP实验或SOPC实验等传统实验平台多数是整体结构型的，虽也可完成多种类型实验，但由于整体结构不可变动，实验项目和类型是预先设定和固定的，很难有自主发挥和技术领域拓展的余地，学生的创新思想与创新设计如果

与实验系统的结构不吻合，便无法在此平台上获得验证；同样，教师若有新的创新型实验项目，也无法即刻融入原本是固定结构的实验系统供学生实验和发挥。因此这类平台不具备可持续拓展的潜力，也没有自我更新和随需要升级的能力。

所以最好的解决方案是：

(1) 在创新实践中，能提供给学生的用于构建任何类型、任何结构和任何规模（几乎）的逻辑资源和存储器资源，丰富到足以涵盖学生的创造力所及的任何形式和规模的设计项目。

所以推荐使用性价比较好的 Cyclone Ⅲ系列的 EP3C55F484，它有 484 脚，BGA 封装；内含 5.6 万个逻辑宏单元（即含 5 万多可供用户使用的 D 触发器，其逻辑资源达到一片 FPGA 中可同时容纳 3 个以上中规模的 32 位嵌入式处理器），240 万 RAM bit 和 4 个锁相环，312 个 9×9 位的高速硬件乘法器等；再外接 32MB SDRAM、250KB SRAM、1GB 并行 Flash、2MB 串行 Flash、16MB 配置 Flash、2GB SD 卡等。这些资源都放置于一块 6 层 PCB 板构成的核心模块板上，即图 F-3 所示的模块板上。

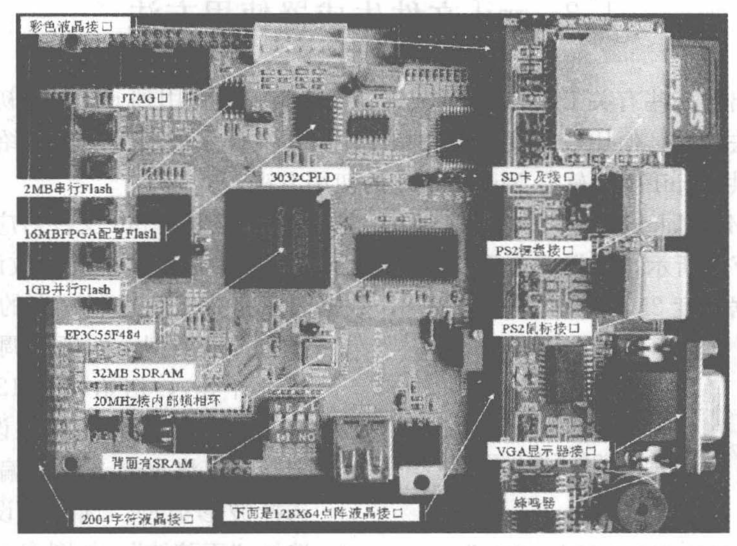


图 F-3 KX-DN8 系统配置的 KX-3C55F+核心板

(2) 在外围接口方面，除大量丰富的接口模块，如 VGA、PS2、USB、SD 卡、RS232 串口、语音处理、AD/DA 等现成的模块外，还提供能适应实验者随时根据自己的创新实验需要，自主安排新功能模块的标准接口。

(3) 将实验硬件平台定位于大容量的Cyclone Ⅲ FPGA，在硬件测试、软件调试、软硬件联合开发与测试方面，甚至包括微指令系统的实时编辑调试中，基于 Quartus Ⅱ平台的强大的测试工具，如 Signal Tap Ⅱ、In-System Sources and Probes 和 In-System Memory Content Editor 等具有不可替代的功能。

KX-DN8 系统很专业地包含诸如基本数字系统设计实验、EDA 技术实验、VHDL/Verilog 硬件描述语言应用实验、SOPC 开发、各类 IP 的应用、基于单片机 IP 核的 SOC 系统实现，以及 8088/8086 IBM 系统核的 SOC 片上系统设计等。

显然，KX-DN8 系列所采用的目前比较流行的模块自由组合型创新设计系统，作为本

教材的实验平台，能较好地适应实验类型多、设计规模大和技术领域跨度宽的实际要求。

KX-DN8 系统的主要特点是：

- 由于系统的各实验功能模块可自由组合、增减，故不仅可实现的实验项目多、类型广，更重要的是很容易实现形式多样的创新设计项目，甚至对于 CPU 这样的大规模逻辑系统的设计。
- 由于各类实验模块功能集中，结构经典，接口灵活，对于任何一项具体实验设计都能给学生独立系统设计的体验，甚至可以脱离系统平台自由组合。
- 面对不同的专业特点、不同的实践要求和不同的教学对象，教师甚至学生都可以自己动手为此平台开发增加新的实验和创新设计扩展模块。
- 由于系统上的各接口，以及插件模块的接口都是统一标准的，因此此系统可以通过增加相应的模块而随时升级。也推荐读者能按照这样的理念，即模块化实验模式的方式自行开发各类模块，构建全新的实验系统。

1.3　mif 文件生成器使用方法

本书中给出的一些有关 LPM RAM 或 ROM 的实验都将用到 mif 格式初始化文件，这可以用不同方法获得，但比较方便的方法是使用 mif 文件生成器。这里介绍康芯公司为本书读者免费提供的 mif 生成软件 Mif Maker 的使用方法。

双击打开 Mif-Maker 2010，如图 F-4 所示。首先对所需要的 mif 文件对应的波形参数进行设置。如图 F-5 所示，选择“查看”，并于此下拉菜单中选择“全局参数设置”。如选择波形参数：数据长度 256，输出数据位宽 8，数据表示格式十六进制（有的情况下需要选择有符号类型，如幅度调制信号发生器的设计等），初始相位 120 度（如设计计 SPWM 中要用到此相位设定），按“确定”后，将出现一波形编辑窗。然后再选择波形类型：选择“设定波形”，再选择“正弦波”，如图 F-6 所示。



图 F-4　打开“Mif _ Maker 2010”

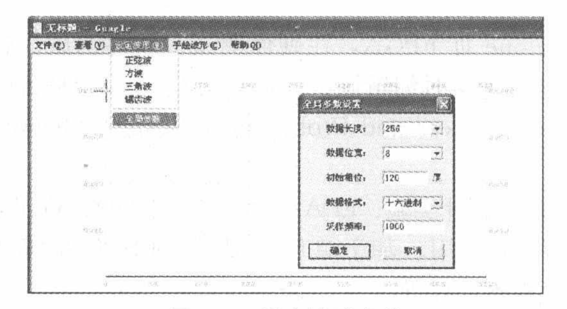


图 F-5　设定波形参数

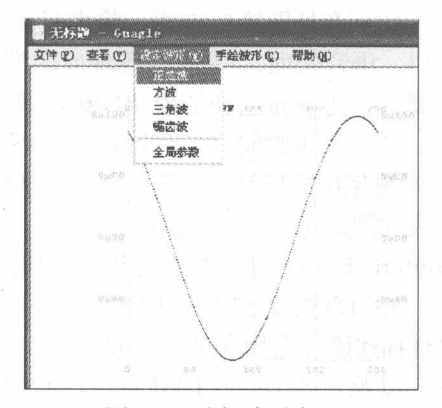


图 F-6　选择波形类型

这时，图 F-6 将出现正弦波形。如果要编辑任意波形，可以选择“手绘波形”项，在下拉菜单中选择“线条”(图 F-7)，表示可以手工绘制线条。然后即可在图形编辑窗中原来的正弦波形上绘制任意波形（图 F-7)。最后选择“文件”中的“保存”，将编辑好的波形文件以 mif 格式保存（图 F-8)，如取名为 WAVE1.mif。

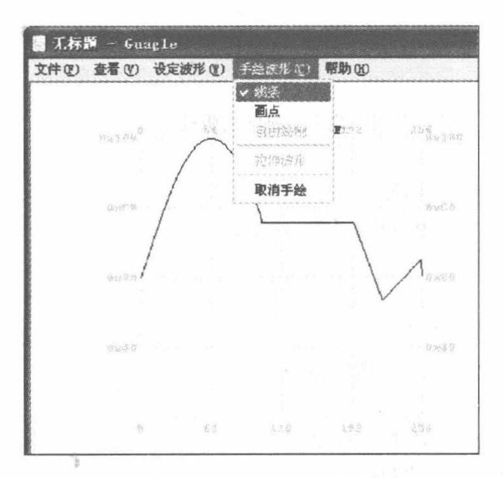


图 F-7 手动编辑波形

图 F-8 存储波形文件

如果要了解编辑波形的频谱情况可以选择“查看”项的“频谱”。如图 F-9 所示的锯齿波的归一化频谱显示于图 F-10 上。

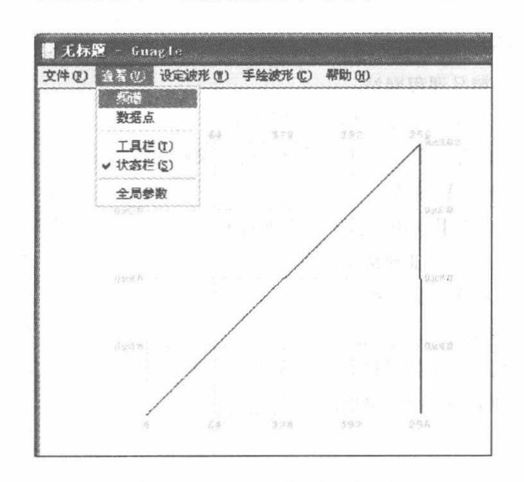


图 F-9 选择频谱观察功能

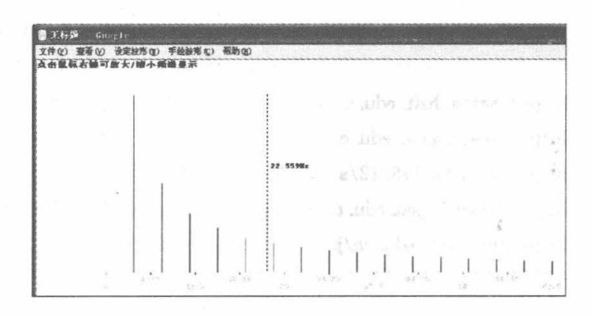


图 F-10 锯齿波频谱